Notes

Jason Reed

March 15, 2007 – February 8, 2009

2007.3.15

Sometimes we take some connective in a position where it is not invertible and by judgmental brute force make it so. For example, taking

$$\frac{\Gamma \vdash C}{\Gamma \vdash C \vee D} \qquad \frac{\Gamma \vdash D}{\Gamma \vdash C \vee D}$$

to

$$\frac{\Gamma \vdash C, D, \Delta}{\Gamma \vdash C \vee D, \Delta}$$

and modifying $\supset E$ to be the modal

$$\frac{\Gamma, C \vdash D}{\Gamma \vdash C \supset D, \Delta}$$

Or for another example, introducing contextual *non*-modal logic to make the implication sequent left rule be something like

$$\frac{\Gamma, [\Psi, A]B \vdash C}{\Gamma, [\Psi]A \supset B \vdash C}$$

and inventing a structural rule

$$\frac{\Gamma \vdash \Psi \qquad \Gamma, A^+ \vdash C}{\Gamma, [\Psi]A^+ \vdash C}$$

Where A^+ is supposed have a positive connective on top, and $\Gamma \vdash \Psi$ is not a classical sequent, but one where the right-hand side is interpreted conjunctively, and the proof-term would be a substitution.

The other case I can think of is the continuation of the first example above of 'classicalization' of disjunction by making implication invertible again on the right, by inventing labels

$$\frac{\Gamma, C[pa] \vdash D[pa]}{\Gamma \vdash C \supset D[p], \Delta} \supset R^a$$

This seems even more exotic than the other two, for it doesn't appear to be simply unpacking the connective as a judgment and permitting whatever derivations happen to be admissible underneath it. However, maybe it could be put in that form, if one thinks of the world-paths as actually being syntactic paths.

The other thought I had is that maybe all of these things are 'fake' or at least unnecessary judgmental innovations if all they signal is a certain kind of focussing discipline. The sequence of asynchronous decompositions and finally application of a structural rule in the case of the contextual left-asynchronous $\supset$ may allow proofs to line up one-for-one with proofs in a system where implication is left synchronous but consecutive synchronous decompositions are required.

I would like to run machine learning algorithms on existing kerning tables as a function of obvious features like smallest interspline distance and area between right and left sidebearing splines.

An old idea: learn a mapping between letters and notes of two durations. An octave from C to C gives me thirteen tones, and two durations gives me twenty-six letters.

2007.3.16

A pursuit game where each player has a 'seeker' and a 'target' which move around on a graph that the player builds during the course of the round. The speed that either item travels along a graph edge is inversely proportional to the edge's length; so that investing a whole bunch of vertices along a line make travel across it faster. Players score points when their seeker is close to their opponent's target.

2007.3.17

Thoughts on Baez's "Categorification"

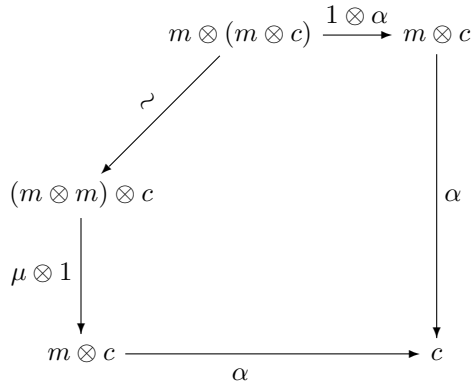
- "Looping" is a name for the process of getting a $k+1$-tuply monoidal $n-1$-category from a k-tuply monoidal n-category by choosing an object of it and index-shifting, getting a new operation.
- The coherence laws for braided (and perhaps plain) monoidal categories *don't* imply that every isomorphism is equal. (!?) This compels me to ask: why are these called "coherence laws" then? Even if that is answered, why do we know we have the right coherence laws?

- A possible answer is that the coherence laws for n-categories arise naturally from thinking about coherence sensibly, and the looping process trashes this property, but nonetheless canonically says what the higher-order laws (whether we call them "coherent" or not) for k-tuply monoidal n-categories should be.

On "Higher-Dimensional Algebra III":

- The "Microcosm Principle" has a formalization! It's something like: 'O-algebra objects can be defined in any 1-coherent O-algebra'. A 1-coherent O-algebra is something like a once-categorified O-algebra.

- Monoid object actions can be defined even for a monoidal category acting on another category. They are referred to as 'riding' this action functor. Say we have an action of a monoidal category $\mathbf{M}$ on a category $\mathbf{C}$. $A : \mathbf{M} \times \mathbf{C} \to \mathbf{C}$. Write this as $(m, c) \mapsto m \otimes c$. So suppose m actually is a monoid object in $\mathbf{M}$. An action of m on $c \in \mathbf{C}$ is a map $\alpha : m \otimes c \to c$ such that



  and similarly for identities.

- Operads are monoid objects in a certain monoidal category.

  Define the SMCC $fam(\mathbf{C})$ 'families in $\mathbf{C}$' to have objects that are sequences of objects from $\mathbf{C}$, and morphisms

$$(x_1, \ldots, x_n) \to (y_1, \ldots, y_m)$$

  consist of specifying a bijection $\sigma : \{1, \ldots, n\} \to \{1, \ldots, m\}$ and a family of morphisms $f_i : x_i \to y_{\sigma(i)}$. The monoidal structure is concatenation of object sequences. This is the free SMCC on $\mathbf{C}$, I think?

  Define the category $prof(\mathbf{C})$ 'profiles in $\mathbf{C}$' to be $fam(\mathbf{C})^{\mathrm{op}} \times \mathbf{C}$.

Define the category $sig(\mathbf{C})$ 'signatures in $\mathbf{C}$' to be $\mathbf{Sets}^{fam(\mathbf{C})^{\mathrm{op}} \times \mathbf{C}}$.

I don't quite see the monoidal structure of $sig(\mathbf{C})$, but it's supposed to have one. Similarly there's allegedly an obvious action from $sig(\mathbf{C})$ to $\mathbf{Sets}^{\mathbf{C}}$, called the 'tautologous action'.

A $\mathbf{C}$-operad is a monoid object in $sig(\mathbf{C})$. An alebra of an operad O is an action of O on some functor $F \in \mathbf{Sets}^{\mathbf{C}}$ that rides the tautologous action mentioned above.

2007.3.18

Liang and Nadathur's "Tradeoffs in the Intensional Representation of Lambda Terms".

- Good source of example higher-order logic programs
- The penalty incurred in deBruijn form of having to renumber things exists, but is not onerous.
- Dependency annotations help with 'eager' implementations, but this advantage is obliterated by moving to a lazy and sharing approach, for which dependency annotations don't seem to help any.

2007.3.19

Reynolds lecture on separation logic

- Separation logic is basically a set of sound rules with respect to the semantics of heaps. Completeness is an impossible goal, just as it is with type systems with respect to program correctness.
- Using stacks of variables is kind of crazy. Aren't they just special bits of heap that are guaranteed to be not addressable?

HOL guy (John Harrison) talk

- Hilbert's 17th problem: a polynomial is positive semidefinite if it's a sum of squares of rational functions.
- We can drop the generalization to rational functions if the polynomial is univariate or a quadratic form. (i.e. every term is exactly degree two)
- The Nullstellensatz: Over an ACF, polynomials p_k have no common solution iff there exist 'cofactors' q^k such that $p_k q^k = 1$.

- The Realstellensatz: Over a real closed field, polynomials p_k have no common solution iff there exist 'cofactors' q^k and s^j such that $p_k q^k + s^j s_j = -1$.

2007.3.20

Realization: trying to do proof irrelevance via focusing as if it were @ doesn't work, because the 'brackets' type operator is bipolar. The elimination rule must blur, and so do a let-binding, and must therefore deal with commuting conversions.

2007.3.21

Fix an injective notion of pairing $\langle , \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Say a model of computation is a partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$. We require axiomatically:

Identity: There exists id_f such that $\forall x. f\langle id_f, x\rangle = x$.

Composition: Given k_1 and k_2, there exists k such that

$$\forall x. f\langle k, x\rangle = f\langle k_1, f\langle k_2, x\rangle\rangle$$

Universality: There is $u_f : \mathbb{N}$ such that $f\langle u_f, x\rangle = f\ x$ for all x.

Say $f \leq_g h$ iff $\exists k. \forall x. f\ x = h(g\langle k, x\rangle)$

Say $f \equiv_g h$ iff $\leq_g$ holds in both directions.

By Identity and Composition, this yields a bicategory for each g. The 1-cells are the ks that witness $\leq_g$. 2-cells are extensional equivalence of g-functions denoted by integers.

Conjecture If $f \equiv_f g$, then $f \equiv_g g$.

2007.3.22

Gustavo talked yesterday about the idea of informal proofs. He made the claim that most published informal proofs are probably correct. I went off on my usual ambiguity-of-measure rant, but there's a separate issue that even apart from that issue, if you look at a single claimed result, the issue of *what* the claim is being made really is is subject to informal understanding. Of course different formalizers will arrive at different proofs, but they will also arrive at different formalizations of the claim.

2007.3.23

From the discussion at Passport last Wednesday, the usual bullshit came out of a discussion on what Art is. For the time being I still cling happily to the Wittgensteinian point of view that there is no essential answer to the question — I have a tendency that I am not particularly proud of (embarrassed by its arrogance) to hope a priori that the course of my life might

follow W's biography and leave me with still greater conceptual reorganizations later in my life.

This point of view could be resummarized as saying, there does exist data out in the world concerning what we *do* call art, though this data is fuzzy, inconsistent, and hard to acquire. Under greater greatest scrutiny, it resolves merely into every instance of something being called art, with no necessary pattern binding it together, and under even greater reductionism, we arrive at the level where it is ambiguous that a person has even uttered the word "art" — for at some stage the slurred sequence of sound had to have historically come from sounds that were decidedly of another language than English, one of its ancestors. Even the *word* 'art' is not immortal, so I am reluctant to believe that its denotation could be.

But apart from the data of what *is* called art, we are asking the question of what *should* be called art. This 'should' — in my most polemic moods, I want to say that it's a complete ghost, that it's not real at all, that there *is no 'should'* to it. But I have to admit that one can get shoulds out of it if one tries.

For example, there is a argument that says this objection could be applied to every word in the language, leaving us with no tools for conceptual discrimination at all! This confuses the claim that 'art' should not mean any particular thing with the straw-man that there is no reason that we should have words to map out those things that we call 'art'. I do find it useful to be able to speak about people attempting to

- reproduce features of the visual appearance of objects in other forms
- evoke emotions
- express ideas
- create visual styles
- violate assumptions

and so forth. And indeed I think these are all interesting activities: but I don't feel worried about which of them is most especially associated with a word whose meaning is *intentionally* undefined.

It's like mathematicians arguing about whether groups are really commutative or not. The answer is, there are groups that are, and groups that aren't.

The notion of 'useful' is pretty nebulous, though. I remember disqualifying uses of it by Donna in reference to personality testing. The thing is that I would like to measure whether one classification scheme is *more useful* than another. It seems certain that having language is more useful

than not, but it seems devilishly difficult — and politically incorrect — to make any comparisons past that point.

On that point I maintain my belief that all garden-variety human languages are in practice essentially equipotent, but in principle there are such things as better- and worse-adapted languages for various communication and cognition tasks. Memorizing words and facts about words (i.e. their 'definitions' as patterns of use) is not *essentially* different from memorizing declarative facts, in that both can help solve real problems, and that some such solutions are more effective than others.

Back to thinking about mathematical formalization: what would I advance as a minimalist notion of proof representation, to maximize clarity of definition, if not of encoding? My first thought is to say: Give me a binary string B and a finite set S of rewrite rules $B_1 \mapsto B_2$. I have a certain sort of strong understanding about the meaning of the question, "can you get to B' from B by using S?" Which is to say, I know what I would accept as an affirmative witness to such a question. I could step-by-step examine each transition, make sure that the antecedent of that step differs from its conclusion only in one segment, and that that segment goes from B_1 to B_2 for some element of S.

However: this setup depends on the integrity of the symbols themselves, and even my understanding of the word "two". If my interlocutor can tell the difference between red zeroes and green zeroes and I am colorblind, I might accept some proofs that she would reject. Conversely, if she can't tell the difference between the symbols at all, she will interpret everything in unary, and accept more proofs than I do.

Furthermore, the 'adequacy proofs' for this system are hard and unreliable. There is just as much opportunity for screwing up the encoding as there is the claim of some theorem. But to speak of adequacy proofs this way presumes there is some external notion of 'what the mathematician *really* means', which I find suspect. Here we come back to the idea that maybe some languages are simply internally *better* work-places, more effective loci in which to develop ideas in the first place.

The meaning of a word is just as disconnected from its sound as the value of a dollar is from its physical representation — although it seems to have inertia, anyway. The difference is that money has only one dimension to float in — inflation and deflation — while words float in a tremendously high-dimensional space. Though maybe it's locally not so high-dimensional?

A Story:

After his wife died, her body received by the moist summer earth, Beu Aiko fell in love with the moon. The moon brought him a gift: it was a small cup, which fit easily in his hand. It felt warm there as he held it, and he saw it was full of light the moon had collected and given to him. She told him to drink it, but he refused, smiling. He put the cup on a shelf in his home, near his oldest books, which he had inherited from his father's mother. The next day, she brought him a larger cup, just as full, and told him to drink it, but he refused again. He hid the cup behind his bed, behind the wool and linen blankets. The cups were made of fired clay, and each bigger than the last: by the seventh day, they had become too large for him to hold with one hand. By the fourteenth day, he struggled to lift them off the ground.

The moon became angry at Beu Aiko, and ceased to love him, but he smiled all the while she screamed at him, and threw at him grass and mud. She left him for several days. When she returned, there appeared another gift from her, next to the house of Beu Aiko. It was a lake, full of the same moonlight she had offered to him before, ripples and whorls writhing almost silently across its surface with each passing breeze.

Beu Aiko ran eagerly to the shore of the lake and splashed toward its center, and there he drowned.

2007.3.24

The feeling of a heavy person sitting next to you on a bus.

The delicate porousness of a banana peel, viewed edge-on; the snap of it as you break the stem.

A culture that has a notion of *gend*, a quality that competes in the same niche as truth. Some people abandon truth as a criterion to be sought out in things and theories and beliefs; they maximize *gend* instead. Gend is an internal quality, and is a notion of internal conformity to principles, just as truth is external. Just because it is internal does not mean it is not amenable to direct observation. But just because it is amenable to direct observation does not mean that observing it is easy without training — just as artists and writers require training to see with precision the truths of the external world.

A breakaway sect that views truth with just as much indifference, but seeks to *minimize* gend instead of maximizing it.

I remember Tobin Coziahr complaining about people who say "I think" and "In my opinion" a lot in their writing, the idea being: you are writing

what you write — of course you think so, and of course it is your opinion. I think the truth is that these are perfectly serviceable markers of reduced confidence, but a person must choose judiciously how often to use them. Used almost incessantly, they theoretically lend infinite confidence to sentences that lack them.

2007.3.25

Ronald Brown, "From Groups to Groupoids": Things get simpler with the Siefert-Van Kampen Theorem if you use groupoids instead of groups. I think it basically claims that π_1 preserves pushouts or something.

2007.3.26

Define $A \mapsto B$, pronounced 'B is a finite version of A' by

$$\frac{}{A \Rightarrow 1 \;\&\; A \;\&\; (A \otimes A) \;\&\; \cdots \;\&\; (A \otimes \cdots \otimes A)}$$

$$\frac{p \Rightarrow A}{p \mapsto A}$$

$$\frac{A \mapsto A' \qquad B \mapsto B' \qquad A' \star B' \Rightarrow C}{A \star B \mapsto C} \; (\star \in \{\rightarrow, \wedge, \vee, etc.\})$$

Conjecture $\Gamma \vdash A$ in ordinary propositional logic iff there exist finite versions of Γ and A such that $\Gamma \vdash A$ in linear logic, translating $\wedge$ freely to $\otimes$ or $\&$.

It seems like I ought to be able to generalize tiling $2n$-gons with rhombuses to something like tiling arbitrary polygons with other arbitrary polygons as long as the total collection of absolute orientations of edges is 'rational' in a suitable sense.

Jared Diamond talk: ugh. Dry, uninteresting, repetitive, bland.

2007.3.27

K. Culik II, "An aperiodic set of 13 Wang tiles" (1996) in Discrete Mathematics 160, pp. 245–251.

Nice, clear paper. Idea: represent a positive real number $r \in \mathbb{R}^+$ as a 'balanced sequence', a map $f : \mathbb{Z} \to \mathbb{N}$ via

$$f(n) = \lfloor (n+1)r \rfloor - \lfloor nr \rfloor$$

Given this, it is possible to make FSMs decorated with input and output symbols along the transitions that multiply balanced sequences by rational

numbers. Executions of these FSMs (which are bi-infinite) form rows of a Wang tiling. The whole tiling consists of an iterated sequence of executions. The machines he use multiply by 3 and 1/2, and so no sequencing of them can ever lead to the identity, yielding a non-periodic set of Wang tiles.

Robert Berger, "The Undecidability of the Domino Problem" (1966) in Memoirs of the AMS 66.

The original settling of the decidability question of Wang tiling.

The argument that Wang and Moore made, contingent on Wang's (refuted) conjecture that any tile set that admits a tiling admits a periodic one, is peculiar. It works like this:

Suppose the plane is not periodically tilable. By conjecture, it is not tilable. Hence there is some finite region (wlog a square) that cannot be tiled.

Suppose the plane is periodically tilable, say with frequency (a, b). Then there is an (ab, ab)-sized torus that serves as a unit cell for tiling the plane.

Start two threads in parallel, checking for each N-square whether it can be tiled as a torus and whether it can not be tiled at all. We are guaranteed evidence either way.

Peter Cho talk: Pretty decent. I guess I confused him with another dynamic typography guy?

2007.3.28

Notes from LF meeting. Topic: **Unification**

$$[\Delta;]\Gamma \vdash U \doteq U'[: V]$$

The Δ is implicit in the implementation, represented by ref cells. The V is also implicit; the Γ is not. We're able to maintain Γ because λs have type labels. Frank says it would have been better to be more bidirectional and keep V, but it's hard to change now.

We would like to maintain the invariants

$$\Gamma \vdash U : V$$

$$\Gamma \vdash U' : V$$

but actually these will fail, because HOU is undecidable. We actually keep track of constraints on the side, and the invariant is, for every substitution that satisfies the constraints, U and U' have the same type after the substitution is applied.

Most the cases are easy, if we mumble about the order of composition of effects or substitutions or what-have-you. Like for instance,

$$\frac{\Gamma \vdash U \doteq U' : \text{type} \qquad \Gamma, U \vdash V = V' : \text{type}}{\Gamma \vdash \Pi U.V \doteq \Pi U'.V' : \text{type}}$$

The hard/interesting case is flex on the left. Faced with $\Gamma \vdash X[\sigma] \doteq U : V$, with $\Gamma \vdash U : V$ and $\Gamma \vdash \sigma : \Psi$ and $X :: (\Psi \vdash V')$ and $\Gamma \vdash V'[\sigma] = V : \text{type}$, then we want to compute a partial inverse substitution σ^{-1} and apply it to U. This is the assignment to X:

$$X \leftarrow U[\sigma^{-1}]$$

Example:

$$u : a, v : a, w : a \vdash X_0\ v\ u \doteq f\ u\ (f\ w\ c) : a$$

This should fail right away because X_0 has no way talking about w. After elaboration and lowering we get

$$X :: ([u :]a, [v :]a \vdash a) \in \Delta$$

and the problem is

$$a, a, a \vdash X[3.2.\uparrow^3] \doteq f \cdot (3; f \cdot (1; c))$$

Note that the spine gets reversed during lowering! The inverse of $[3.2.\uparrow^3]$ is $[\_.2.1.\uparrow^2]$. This breaks on the 1. The one-sided substitution inverse is just matrix transpose, isn't it?

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{matrix} 3. \\ 2. \\ \uparrow^3 \\ \\ \\ \vdots \end{matrix}$$

gets mapped to

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{matrix} \_. \\ 2. \\ 1. \\ \uparrow^2 \\ \\ \\ \vdots \end{matrix}$$

and $AB = I$, (think of matrices acting on row vectors on their left) though $BA \neq I$.

Now if we find during application of an inverse substitution σ^{-1} to an evar closure like $Y[M.N.\uparrow^n]$ that $\sigma^{-1}N$ doesn't exist, we need to prune Y.

Now, consider non-pattern equations. We only ever postpone equations like $X[\sigma] = U$, so these are bundled up with the evar X itself. Pro: whenever we instantiate an X, we know what constraints are associated with it! Con: when unification returns 'true', it doesn't necessarily mean that unification worked — we have to dig through the term to find the variables that might still have constraints left.

Lie: if constraints are left, they cannot be solved by pattern unification. **But** sometimes things cannot be solved by pattern reconstruction, and nonetheless their constraints are eliminated, specifically in the case of like $X\ M \doteq X\ M$ — there is a special hack to solve this.

The difficult part is this: suppose at the source level we have

$$X\ u \doteq f\ u\ (Y\ (Y'\ u\ w))$$

Do we prune the second argument of Y', or the first argument of Y? If we just suspend, we lose the information that X starts with f. So what we do is

$$X \leftarrow \lambda u.f\ u\ (Z\ u)$$
$$Z\ u \doteq Y\ (Y'\ u\ w)$$

But what type does Z have? On a bad day, its type could involve X, even. Here is where we would have loved to keep track of V.

2007.3.29

A game where you have to arrange an org chart for a company to maximize something. Imagine there are hidden variables like "personality" of employees such that any collection of tree-siblings that contains an opposing pair results in 'conflict' and they don't get any work done.

'Idea' tokens (maybe good ideas, maybe bad ideas?) are emitted from the leaves and filter up through the hierarchy. Some people are good idea generators, some people are good idea filters.

'Managers' may reject good ideas, reject bad ideas, turn bad ideas into good ideas.

2007.3.30

Reading a paper by Thomas Erhard and Laurent Regnier, titled 'The Differential Lambda-Calculus' (2003)

Since they deal with R-linear combinations of terms, there is a term 0 that is like Girard's daemon. The basic idea I think is that if we had $\otimes$ pairs, then

$$\frac{\partial}{\partial x}(M \otimes N) \cdot u = (\frac{\partial M}{\partial x} \cdot u) \otimes N + M \otimes (\frac{\partial N}{\partial x} \cdot u)$$

Their typing rules say coalescingly that

$$\frac{\Gamma \vdash s : A_1 \to \cdots \to A_n \to B \qquad \Gamma \vdash u : A_i}{\Gamma \vdash D_i s \cdot u : A_1 \to \cdots \to A_n \to B}$$

But I would have expected maybe something like

$$\frac{\Gamma \vdash s : A \to B}{\Gamma \vdash Ds : A \to A \multimap B}$$

In the other extreme, making things as judgmental as possible, I might have expected

$$\frac{\Gamma, x : A; \Delta_1 \vdash s : B \qquad \Gamma; \cdot \vdash t : A \qquad \Gamma; \Delta_2 \vdash u : A}{\Gamma; \Delta_1, \Delta_2 \vdash (Dx.s) \cdot (t, u) : B}$$

where Δ is a collection of linear hypotheses. Is this rule conservative? Can I prove $(A \to B) \to (A \to A \multimap B)$ ordinarily?

$$\frac{XXX}{\dfrac{A \vdash !A \qquad B \vdash B}{A \to B, A; A \vdash B}}$$

No! I would need something like $A \otimes !A \vdash !A$. Does this affect my LLF encoding? Are there terms of type $(A \to B) \to (A \to A \multimap B)$?

$$f : A \to B, x : A, \alpha : w, y : A @ \alpha \vdash f\ y : B[\alpha]$$

No! f would only accept an argument at ϵ.

2007.3.31

Regarding the translation from the refinement calculus to proof irrelevance: Do you really need to allow dependently-sorted 'classifiers' when declaring refinement sorts of type families?

2007.4.1

So each base refinement $s :: L \subseteq K$ translates to a predicate on things of type $a \cdot S$ for some S. We need to kind of code up Σs, but also to code up proof irrelevance on the right. The thing we ought to be able to do is just yield proofs of the type 'for real', and only ever require them irrelevantly.

2007.4.2

Reading `http://worrydream.com/MagicInk/`.

> Command-line systems are criticized for forcing the user to learn the computer's language. Modern GUIs may be easier

> to use, but they are not much different in that respect. The GUI language consists of a grammar of menus, buttons, and checkboxes, each labeled with a vocabulary of generally decontextualized short phrases. The user 'speaks' by selecting from a tiny, discrete vocabulary within an entirely fixed grammatical structurea bizarre pidgin unlike any human language, unexpressive and unnatural.

This is a peculiar perspective: why is the set of interactions with a computer considered a horribly broken, cobbled 'language', and the set of interactions with other machines (say, a car) considered just that, a set of ways that one can interact with it?

Tangible Functional Programming, Conal Elliott, is pretty interesting.

2007.4.3

More from "Magic Ink":

> Prominent usability pundits have claimed that the public is becoming more discriminating, but since this claim underlies their consultancies' sales pitch, it is far from an unbiased observation. I see the oppositeas technology races ahead, people are tolerating increasingly worse design just to use it. The most beautifully-designed DVD player will go unsold if the competition costs the same and has S-Video output, or plays MP3s from memory sticks. Good design makes people happy, but feature count makes people pay.

This sounds like a unit confusion similar to nature versus nurture. Prettiness of design and number of features are basically incommensurable things; when we ask a question like "if I pump $X into the design department, or into the feature-making department, will I get better results?" we have asked something meaningful, but we are no nearer to an answer to that question if we just sit and navel-gaze about whether features or design are intrinsically more important.

Some eariler thoughts: It is important to keep in mind that this guy's claims are not likely valid for software that is not so-called 'information software'. Some of the difficulties of designing this sort of software arise because our expectations are higher for software — we want to believe that adding context or interactivity or something actually makes the system *better than* paper. We *could* simply make with software what we made with hand, on paper, and that would be no more difficult to achieve than the results we achieved by hand, on paper. But it would also be no better.

I like the attention to what questions the user probably wants answered — but his guesses are not always the same as mine! In his Amazon redesign, the price of a book is still rather small, but it's a piece of information that is still rather important to me.

The unobtrusive hyper-link anchor sharps are *fantastic*. Mostly invisible, quiet and gray when they appear, appearing whenever you mouse-over the *paragraph* not just the region where the sharp itself is, thereby indicating what it's attached to, and having a text suggestive of their purpose if you know HTML. The only problems I have detected are that the sharp disappears between the paragraph and the sharp, and that the paragraph numbers seem to skip around occasionally.

In paragraph 10 he gets pretty ballsy asserting that all HCI textbooks on the market are crap. Myself I feel pretty reluctant when it comes to related work sections saying definitively that nobody has successfully done a particular thing.

Paragraph 18 contains (at least the beginnings of) a pleasingly enlightened understanding of the continuum of data and programs — but I don't agree with p19 necessarily.

At about paragraph 265 now.

Reading William's notes on LF plus refinements. Quite pretty. The main question that occurs to me is that in the specification of atomic refinements of type families, why is it that Π-elimination takes an argument, and &-elimination produces no term part?

2007.4.4

Ok, so some notes on proof irrelevance.

- Example of composite numbers.
- Example of strict lambda terms.

We have a failure to express the types we really mean. It's true that there are workarounds, but so too there are workarounds for a system that doesn't have, say, higher-order abstract syntax. So let's explore a type-theory that lets us make more intrinsic encodings of these sort of phenomena.

What we want to do is make some arrows and applications 'proof-irrelevant' like

```
comp/ :  ΠN : nat.  is-comp N -:> comp.
```

so that these two guys are in fact considered equal:

```
six-comp1 :  comp = comp/ six o (is-comp/ two-by-three).
six-comp2 :  comp = comp/ six o (is-comp/ three-by-two).
```

So let's think intuitively about what this function type is supposed to mean, before sketching out the formalism.

$A \to^{\div} B$ is a function that requires evidence that A is inhabited, but the B it produces is guaranteed not to depend on which A is provided. This is the contract we expect to be satisfied when we declare constants with irrelevant function types. We'll cook up the notion of equivalence of canonical forms so that these are really equal.

But let's make sure this type live in our theory as a first-class function type: we'll be able to write our own irrelevant lambda expressions, too. This means we'll have a new sort of hypothetical judgment $x \div A$. It is an assumption that A is inhabited, but we are prohibited, when we build terms from x, of depending on the identity of x.

And so we'll need a new substitution principle for these hypotheses. What terms M can we safely substitute for $x \div A$? Since x exists in some environment where its 'client' promises not to care about its identity, we can be *more* reckless in building a term to substitute for x — in particular we can use irrelevant hypotheses in building up M. Let $\Gamma^{\oplus}$ mean Γ with all $\div$ switched to :. Then the new substitution principle is if $\Gamma, x \div A \vdash N : B$ and $\Gamma^{\oplus} \vdash M : A$, then $\Gamma \vdash [M/x]N : B$.

Ok, formalism.

Syntax

$$\begin{aligned} \text{Normal Terms } M &::= \lambda x.M \mid R \\ \text{Atomic Terms } R &::= R\ N \mid R \circ N \\ \text{Atomic Types } P &::= a \mid P\ M \\ \text{Types } A &::= P \mid \Pi x \star A.B \end{aligned}$$

Equality

$$\frac{R \equiv R'}{R \circ N \equiv R' \circ N'} \qquad \frac{R \equiv R' \qquad N \equiv N'}{R\ N \equiv R'\ N'}$$

Judgments ...

Typing

$$\frac{\Gamma, x \star A \vdash M : B}{\Gamma \vdash \lambda x.M : \Pi \star A.B : \text{type}}$$

$$\frac{\Gamma \vdash M : \Pi x{:}A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : [N/x]B}$$

$$\frac{\Gamma \vdash M : \Pi x \div A.B \qquad \Gamma^{\oplus} \vdash N : A}{\Gamma \vdash M \circ N : [N/x]B}$$

$$\frac{\Gamma A : \text{type} \qquad \Gamma, x \star A \vdash B : \text{type}}{\Gamma \vdash \Pi x \star A.B : \text{type}}$$

2007.4.5

Sean talked about decision procedures for the first-order theory of reals. Cohen's algorithm works by building up sign matrices, which are somehow computed recursively in terms of the derivatives of polynomials. It's somewhat amazing to me that everything reduces to polynomials in the first place.

2007.4.6

> Il n'y a qu'un cas où une œuvre ne vaut rien: c'est quand elle correspond aux intentions de l'auteur.
>
> *(In only one case is a work worth nothing: when it corresponds to the intentions of the author.)*
>
> J. L. Borges, as quoted in *Esthétique de L'Oulipo*

> On préfèra dans cet essai adopter une définition pragmatique, empruntée à Umberto Eco. Elle a l'avantage de clarté et de la simplicité: « La littérature, c'est la littérature. » L'idée semble tautologique, elle ne l'est pas tant que ça. Se plaçant résolument du côté du lecteur (après tout, il n'est pas de littérature sans lecteur), évoquant la « tradition littéraire », Eco y place l'ensemble des textes « produits à des fins non pratiques (comme le serait tenir des registres, citer des lois et des formules scientifiques, enregistrer des procès-verbaux de séances ou fournir des horaires de chemins defer) mais plutôt *gratia sui*, par amour d'eux-mêmes — et que l'on lit pour le plaisir, l'élévation des connaissances, voire comme pass-temps, sans que personne ne nous y contraigne (exception fait des obligations scolaires) ».
>
> Que voilà une vision latine et réjouissante, qui s'oppose hardiment au pragmatisme commercial de certain libraries anglo-saxonnes! Celles-ci divisent la littérature en deux rayons distinct: *Fiction*, pour ce qui est jugé divertissant, *Litterature* pour le reste. Les lecteurs ne risquent pas ainsi l'erreur tragique d'acheter un Calvino ou un Borges pour se distraire.
>
> *Esthétique de L'Oulipo* p.45

2007.4.7

Still having a hard time breaking through some invisible conceptual barrier thinking about the foundations of mathematics.

It has some relationship to a lesson I am trying to take away from Wittgenstein: the most general lesson is that *there is such a thing as a bad question*. The more specific versions involve certain forms of questions. A common one is "What is X, really?" It's somehow the most anti-E-prime thing you could ask.

Here I'm reminded of positivism: because I want to say that in order to make a question definitely a good question, you should have some notion of what counts as evidence for or against it ahead of time. This is some kind of condition on *definiteness* of the question.

There's a background assumption about words that they may be "specific" or "vague", and that something like monosemy actually exists. I am reluctant to accept this. I think words' denotations are best thought of as probabilistic smears of situations in which they are used, and so there is no such thing as a word that means *one* thing, because there is no good notion atoms in this space of contexts — a word that is only usable in precisely one situation seems like a useless corner case.

However, denotations can be more or less *crisp* at their edges, and this something about is what I mean by definiteness. There is the (apparently!) separate issue of reliability in transmitting such a concept to another person, but I think there's something deeply similar about transmitting a concept to another person to transmitting it to your own future self — I need to go back and look at Wittgenstein on private languages, because I think the knot might get untied there.

2007.4.8

Sean McLaughlin explained to me more about resistor networks. They seem pretty neat. The underlying connection between them and random walks is that they both satisfy what I feel tempted to call a 'balance' constraint, even though I am not sure it is the same thing as the notion of 'detailed balance' I remember from the Metropolis algorithm. In the unlabelled undirected graph case it's just that every (non-boundary) vertex's value is the average of its neighbors. That this follows from Kirchoff's and Ohm's laws in the case of voltages is pretty easy to see, and in the random walk case it's about as trivial.

I made an error, though, thinking that the resistance of an edge should correspond to the number of *parallel* edges that should represent it in an unlabelled (but multi-edge-having) graph: obviously it should be a series of consecutive edges to model a high resistance.

2007.4.9

Reading the current episode of Baez's "Weekly Finds", week249.

Lemma 0.1 *Suppose G acts transitively on X. Pick $x \in X$. There is a*

bijection $f : X \cong G/Stab(x)$ *given by* $f(gx) = [g]_\sim$.

Proof First of all, having $y \in X$ uniquely determines a g such that $gx = y$, and we have existence of g because the action is transitive. The map is obviously surjective, because for any $[g]_\sim$, the element $gx \in X$ maps to it. Now to show injectivity. Suppose we have $[g]_\sim = [h]_\sim$, that is, $f(gx) = f(hx)$. We're to show $gx = hx$. Because $g \sim h$, there is $k \in Stab(x)$ such that $gk = h$. Thus $hx = gkx = gx$. ■

2007.4.10

A 'syzygy' is a kind of 2-cellish relation between relations?

2007.4.11

Is it really true that any well-typed set of clauses for a refinement of a type family that happens to be, say, an intersection can adequately be erased to clauses for the maximal refinment of it?

2007.4.12

Can unification be done the hybrid system in a well-moded way involving worlds?

2007.4.13

- One reason why moral judgments are difficult. In principle it is easy to persuade well-meaning people to do arbitrarily atrocious things: you need only make them not believe that they are doing so. People play video games and honestly have no qualms about 'killing' their opponents, because they equally honestly believe that they are not causing any actual death or (significant) suffering.

  I don't believe that I am harming a rock when I kick it. Indeed, if I were to believe that a rock *had* moral rights, I would have to determine which things the rock 'wanted'! Our recognition of things around us as having agency and deserving protection goes hand in hand with our beliefs about which states they want to be in, or otherwise normatively *should be in*, on what counts as external evidence of suffering or happiness and so on.

- There is nothing that I have 'always' done. There are things that perhaps I have done each of the finite times I made a certain choice, but I should not underestimate the *finiteness* of this sample. I have not always liked or always disliked anything — I have done so over a certain period. However, there are things that I have *never* done, meaning that without quantificational care, always and never are not complementary.

- Here is a certain facet of the idea of trust: I might feel very comfortable sharing my arbitrariest thoughts with another person, mentioning them to her with the same candor as I might ruminate by myself — except, clearly, there are some thoughts that meet with resistance even in the space of my own brain. In these situations, I am acting less than whole, being reluctant to loose a confession from one part of my self to my inner critic.

  But, really, what cause do I have to be ashamed of myself, apart from the pragmatic end of antireinforcing bad behavior? To move forward I have to achieve some level of dispassion, and draw some circle within which I can assess what has happened to me and what I have done without internal censorship.

2007.4.14

Read some of Kirby, Dowman and Griffiths's "Innateness and culture in the evolution of language". I am disappointed that their model depends on a mapping from 'meanings' to symbols, and not just on the behavioral use of symbols.

2007.4.15

It is amazing how the tiniest quantities of silence can have linguistic meaning — that a statement comes slightly later than expected, makes it seem dishonest in the right context. Bounded rationality in a linguistic setting should account for this, that communicating agents know *that* evidence of the other agents using extra computational resources means something.

2007.4.16

Consider the problem of unification with labels. The basic query is

$$\Delta; \Gamma \vdash M \doteq N \Leftarrow A[p]$$

But this will kick out residual constraints of basically the same form. The role of the labels is to ensure compatibility with the interpretation of linear logic. I want to be able to detect which clauses can actually generate terms that are not only well-typed, but well-labelled.

$$\frac{\Delta; \Gamma, x : A \vdash M \doteq N \Leftarrow B[p]}{\Delta; \Gamma \vdash \lambda x.M \doteq \lambda x.N \Leftarrow \Pi x{:}A.B[p]}$$

$$\frac{X :: (\Psi \vdash B[q]) \in \Delta \qquad \Delta; \Gamma \vdash \sigma : \Psi}{\Delta; \Gamma \vdash X[\sigma] \doteq M \Leftarrow A[p]}$$

2007.4.17

Here is the units problem with the nature vs. nurture argument. The most apt clarification I can think of right now of the idea "nature" is the space of possible genomes an organism might have, together with a metric of similarity between them. Similarly, "nurture" might as well refer to the possible environments an organism might be born into and live in. The question "which is a more important determiner of some observable trait, nature or nurture?" then translates into a question something like "is the derivative of this function bigger in one variable, or another?" but the problem with asking this question is that the two variables concerned have different units. I do not know a priori whether one base-pair change 'counts as' the same as, less than, or more different than being raised by a family that makes, say, \$10k more per year.

And this objection is only made after assuming that these spaces are easily quantifiable! It is not at all clear that every base pair is equally significant, and the space of different environments does *not* in fact present itself as any clear one-dimensional quantitatively indexed structure.

Words are pagan gods, and sentences their mythology: we engage in syncretism every time we translate. When we say English *word* is French *mot*, we do the same thing as when we say Greek *Zeus* is Roman *Jupiter*.

Habits are important. Learning is slow, but eventually effective. Writing things down is importnat.

2007.4.18

In the proposed LF module system, it is uncertain whether one should allow mixing-in of ELF-like constructs like `%solve`. On the one hand, they seem useful to allow logic programming over abstract signatures, and yet they are 'effectful' in that they might not terminate, and are peculiarly 'early-binding' in contrast to the intended behavior of the other metatheory checks.

2007.4.19

What is important is not how data is represented, but what interface it satisfies.

What interface is provided by weak n-categories? In a set, I can ask whether elements are equal. In a category, I can ask for the hom-set that exists between them. This gives me a new set of answers; the hom-set could be empty, or it could be populated (but not contain an isomorphism), or it could contain an isomorphism.

Is some doctrine of ω-groupoids with structure enough to describe ω-categories?

2007.4.20

The problem of words and rules that Wittgenstein raises and which Kripke focusses on is quite difficult to even nail down as a problem. I want to ask, what is the difference between rote memorization and rule learning? I must instead apparently ask, *is* there a discernible difference between rote memorization and rule learning? For even to be able to respond with the *same* response at *identical* stimuli, I must have some notion of equality of responses and stimuli. This is unavoidable, for I can conceive of a notion of equality that is so discriminatory as to be useless, one that says any two responses (or stimuli) at different times are necessarily disequal. One *must* have a notion of 'transport' of behaviors across time at a minimum (and very probably across agents, and across space) to make anything sensible come out of a theory.

2007.4.21

Dependent types seem to be just as much a notion of refinement as anything else: the intrinsic *structure* of LF terms, sufficient to compare them for equality and carry out substitution, seems totally determined by positing one base type, and the type constructor $\rightarrow$. This is not, the case, however, for proof irrelevant terms or singleton types. In the case of proof irrelevance, instead of speaking of type-directed equality, I just demand that the syntax reflect the types in certain ways: irrelevant arrow is treated as a genuinely different type, because it has different equality behavior.

But does this refute my earlier claim that I never need to think of dependent types as anything *but* refinements? I think it might. For if an irrelevant arrow allows any term at type o to occupy its argument, and considers all of them equal, then isn't a term well-typed provided there is an inhabitant of o (which there very well might be, in context) even if there is *not* a term of the appropriate refinement? (since I can't equationally distinguish them?)

No, I don't think this refutational argument really works. I'm treating irrelevant equality as a supervenient equivalence relation anyhow, so that I can formulate a type-checking algorithm that *can* inspect irrelevant arguments in order to be decidable. Nonetheless the type structure requires the terms to be at least *marked* with irrelevant applications so that this definition of irrelevant equality can get a foothold.

What about singleton types? Do they have terms with nice canonical forms? Or is the point that interpreting a term at different types gives different supervenient equivalence relations on terms, and canonical forms remain the way they are?

I perceive some duality between refinements as 'after-the-fact' supervenient *subsets* of existing types (think: equalizers) and proof irrelevance and singleton types as 'after-the-fact' supervenient *quotients* of existing types (think: coequalizers).

Here is a recurring idea: one should not ask what a word means, but what meanings are available, and which observable things happen.

Worse: "What does art mean?" Better: "People have made images using paint, and pencils, and computers. It is a hard but learnable skill to make these images in various styles. There are mappings from the human visual system's inputs, and to such images, such that it forms a sort of one-sided inverse.

People have made objects by sculpting clay and marble. People have made objects by gluing together objects, welding, soldering, arrangement.

Some of the above creations seem pleasing in various ways to various people: they cause calm, laughter, excitement, curiosity."

2007.4.22

1000 Blank White Cards is a very 'flat' game. Cards occasionally interact with one another, but rather rarely. What sort of cultural or game-regulatory phenomena would change this to make it 'deeper'?

2007.4.23

Can HOU be done effectively without maintaining types (or labels) for anything but unification variables? I think it would only be during actual inversion of pattern substitutions that one would need to schlep around types.

2007.4.24

Nicola Gambino had a talk about the connection between identity types in Martin-Löf type theory on the one hand, and the theory of weak n-categories on the other.

Apparently one can interpret a type theory into the 2-category **Grp** of groupoids, functors, and natural transformations to refute the admissibility of the uniqueness of identity proofs (UIP) rule

$$\frac{\Gamma \vdash M, N : A \qquad \Gamma \vdash P, Q : Id_A(M, N)}{\Gamma \vdash new(M, N, P, Q) : Id_{Id_A}(P, Q)}$$

and maybe also groupoid semantics can be used to refute the admissibility of the 'reflection' rule

$$\frac{\Gamma \vdash P : Id_A(M, N)}{\Gamma \vdash M = N : A}$$

though I wasn't entirely clear on that point.

2007.4.25

Todd Wilson showed that it is sometimes useful to use Church encoded types as indices for other types; you can get finite types that a more full range of functions.

2007.4.26

Hypocrisy is fundamental, and the golden rule a social construction, which can only arise *after* we recognize some subsets of the universe as *not I* but *morally equivalent to I*.

2007.4.27

Listening to a talk by Benjamin Pierce.

```
get : C -> A
put : A -> C -> C

put (get c) c = c
get (put a c) = a
```

He says if you add an extra rule that says two puts in sequence are equivalent to the second, then in fact C must be isomorphic to some product $A \times B$.

Well-behavedness of get and put with respect to list data can be phrased in terms of equivariance up to some equivalence relation, such as reorderability of lines.

2007.4.28

I am unsure whether this termination problem is related to the one Frank mentioned.

Start with the unification equations

$$(X[1] \doteq c \cdot Y[Z[2]]) \wedge (Y[1] \doteq X[Z[2]])$$

in the context $\Gamma = o, o$. The types of the existential variables are all $o \vdash o$, and the type of c is $o \to o$.

Focus on the first conjunct. Though we know at least one of Y, Z projects out its argument, we don't know which of them does. However, we do know X's instantiation must start with a c, though. So make up a new variable $X' : o \vdash o$, instantiate $X \leftarrow c \cdot X'[id]$, and add the constraint $X'[1] \doteq Y[Z[2]]$ to obtain

$$(X'[1] \doteq Y[Z[2]]) \wedge (Y[1] \doteq c \cdot X'[Z[2]])$$

which is just an α-variant of the original problem.

2007.4.29

Thinking about inverse substitution.
Judgment $M[\sigma]^{-1} = N$.

$$\frac{}{\underline{\mathsf{n}}[\underline{\mathsf{n}}.\sigma]^{-1} = \underline{1}}$$

$$\frac{\underline{\mathsf{n}}[\sigma]^{-1} = \underline{\mathsf{i}}}{\underline{\mathsf{n}}[\underline{\mathsf{m}}.\sigma]^{-1} = \underline{\mathsf{i}+1}}\, n \neq m$$

$$\frac{\underline{\mathsf{n}}[\sigma]^{-1} = \underline{\mathsf{n}'} \qquad \rho[\sigma]^{-1} = \rho'}{(\underline{\mathsf{n}}.\rho)[\sigma]^{-1} = \underline{\mathsf{n}'}.\rho'}$$

$$\frac{}{\uparrow^{m+n}[\uparrow^n]^{-1} = \uparrow^m}$$

$$\frac{\uparrow^n[\sigma]^{-1} = \uparrow^i}{\uparrow^n[\underline{\mathsf{m}}.\sigma]^{-1} = \uparrow^{i+1}}$$

This seems to be unable to compute $id[\underline{1}.\uparrow]^{-1}$ however. Is this actually a problem?

2007.4.30

Sapir-Whorf says the things we can think are in part determined by what linguistic structure is available — but this seems almost like a tautology if we consider both 'the things we can think' and 'what is available' to be essentially linguistic (and ultimately behavioral) *habits*.

I appreciate more and more how critical the idea of 'following rules' is to Wittgenstein's linguistic claims.

Doug Hofstadter started writing GEB when he was just a little older than me.

Suppose I want to set up a variable $X : A$ where A mentions X somewhere. Say A is forced to be $\cdot \vdash a \cdot (c \cdot X)$.

```
o : type.
a : o -> type.
c : a M -> o.
d : {X : a M} a (c X) -> type.
- : d X X.
```

2007.5.1

At the discussion about Girard, Lafont, Taylor's book (which I keep thinking of as essentially Girard's book, given the eccentricity of the writing) William asked some question about what essential role equality plays in 'canonical forms' LF. Somehow I had the notion that what really matters is how many equivalence classes there are (and how they behave) not how many elements each equivalence class is supposed to have — however, as an implementation detail, the how 'many elements' question may be of some importance.

A simple formal question: is the category of coherence spaces equivalent (isomorphic?) to the category of graphs?

2007.5.2

On labelled unification:

I need to work out how much typing (and labelling) information needs to be around. There are probably contextual *label* variables as well as term variables. Given that term contextual variables almost certainly need labels, they must have contexts with label variables as well. Are label evars introduced any more freely than other variables? Does this only arise during inversion?

I think the termination argument is that all such equations *can* be postponed until the end, but perhaps one does not want to in practice.

Is there a set of good rewrites that is nonetheless terminating? Eliminating a variable is progress. The counterexample for regular LF unification shows that 'almost eliminating' a variable by inferring that it must contain some part is not actually a simplification, in that it neither reduces the variable count nor the dependency count of some variable.

William successfully convinced me that the following 'semantic' notion of subtyping is at least worth paying attention to:

$$(S_1 \leq S_2 :: \tau) := (x :: S_1 \vdash \eta_\tau(x) :: S_2)$$

because it makes proving completeness of some axiomatization of subtyping feasible. Something still sits ill with me with the idea of saying that subtyping is 'essentially about' such an identity theorem, though, since it seems too tied up with eta expansion, a process that by itself, apart from subtyping, has a notion of correctness. Value inclusion

$$\Gamma \vdash M :: S_1 \Rightarrow \Gamma \vdash M :: S_2$$

still *feels* like what I mean by subtyping.

2007.5.3

Two fundamental, recurrent conflicts: One, language does not afford precision, but mathematics *seems to* attain it. Two, language is conventional and arbitrary, and yet our success in inferring rule-following behavior in other agents is deeply dependent on the assumption that their cognitive machinery is like ours.

2007.5.4

Contextual modal type theory seems to be splittable into contextual types, and modal types. Contextual types are just iterated function spaces associated (and perhaps commuted) another way round. Likewise single-step function spaces where the domain is a big sigma.

2007.5.5

Thinking about base-type polymorphism in LF.

$$\begin{array}{rrcl} \text{Heads} & H & ::= & c \mid x \\ \text{Base Classifiers} & v & ::= & H \cdot S \mid \text{base} \\ \text{Classifiers} & V & ::= & \Pi x{:}V_1.V_2 \mid v \\ \text{Terms} & M & ::= & \lambda x.M \mid H \cdot S \\ \text{Spines} & S & ::= & () \mid (M;S) \end{array}$$

$$\frac{H : V_1 \in \Gamma \cup \Sigma \qquad \Gamma \vdash S : V_1 > V_2}{\Gamma \vdash H \cdot S \Rightarrow V_2}$$

$$\frac{\Gamma, x : V_1 \vdash M \Leftarrow V_2}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}V_1.V_2}$$

$$\frac{}{\Gamma \vdash () : V > V} \qquad \frac{\Gamma \vdash M \Leftarrow V_1 \quad \Gamma \vdash S : [M/x]^{V_1} V_2 > V_3}{\Gamma \vdash (M;S) : \Pi x{:}V_1.V_2 > V_3}$$

$$\frac{\Gamma \vdash M \Rightarrow v \qquad v = v'}{\Gamma \vdash M \Leftarrow v'}$$

$$\frac{\Gamma \vdash H \cdot S \Rightarrow \text{base}}{\Gamma \vdash H \cdot S \Leftarrow \text{ok}} \qquad \frac{}{\Gamma \vdash \text{base} \Leftarrow \text{ok}}$$

$$\frac{\Gamma \vdash V_1 \Leftarrow \text{ok} \qquad \Gamma, x : V_1 \vdash V_2 \Leftarrow \text{ok}}{\Gamma \vdash \Pi x{:}V_1.V_2 \Leftarrow \text{ok}}$$

Conjecture If $\Gamma \vdash M \Leftarrow V$ and $\Gamma, x : V, \Gamma' \vdash J$, then $\Gamma, \sigma\Gamma' \vdash \sigma J$, where $\sigma = [M/x]^V$.

$$\begin{array}{rlll} \text{Heads} & H & ::= & c \mid x \\ \text{Base Classifiers} & v & ::= & H \cdot S \mid \text{base} \\ \text{Classifiers} & V & ::= & \Sigma x{:}V_1.V_2 \mid \Pi x{:}V_1.V_2 \mid v \\ \text{Terms} & M & ::= & \langle M, N\rangle \mid \lambda x.M \mid H \cdot S \\ \text{Spines} & S & ::= & (\pi_i; S) \mid (M; S) \mid () \end{array}$$

$$\frac{\Gamma, x : V_1 \vdash M \Leftarrow V_2}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}V_1.V_2} \qquad \frac{\Gamma \vdash M \Leftarrow V_1 \quad \Gamma \vdash S : [M/x]^{V_1} V_2 > V_3}{\Gamma \vdash (M; S) : \Pi x{:}V_1.V_2 > V_3}$$

$$\frac{\Gamma, x : V_1 \vdash M \Leftarrow V_2}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}V_1.V_2} \qquad \frac{\Gamma \vdash S : V_1 > V_3}{\Gamma \vdash (\pi_1; S) : \Sigma x{:}V_1.V_2 > V_3}$$

$$\frac{\Gamma \vdash R \Rightarrow \Sigma x{:}V_1.V_2}{\Gamma \vdash \pi_2 R \Rightarrow [\pi_1 R/x]^{V_1} V_2}$$

$$\frac{H : V_1 \in \Gamma \cup \Sigma \qquad \Gamma \vdash S : V_1 > V_2}{\Gamma \vdash H \cdot S \Rightarrow V_2}$$

$$\frac{}{\Gamma \vdash () : V > V}$$

$$\frac{\Gamma \vdash M \Rightarrow v \qquad v = v'}{\Gamma \vdash M \Leftarrow v'}$$

$$\frac{\Gamma \vdash H \cdot S \Rightarrow \text{base}}{\Gamma \vdash H \cdot S \Leftarrow \text{ok}} \qquad \frac{}{\Gamma \vdash \text{base} \Leftarrow \text{ok}}$$

$$\frac{\Gamma \vdash V_1 \Leftarrow \text{ok} \qquad \Gamma, x : V_1 \vdash V_2 \Leftarrow \text{ok}}{\Gamma \vdash \Pi x{:}V_1.V_2 \Leftarrow \text{ok}}$$

Here's a try not in spine form:

$$\begin{array}{rlll} \text{Base Classifiers} & v & ::= & \text{base} \mid R \\ \text{Classifiers} & V & ::= & \Pi x{:}V_1.V_2 \mid v \\ \text{Terms} & M & ::= & \lambda x.M \mid R \\ \text{Atomic} & R & ::= & x \mid R\ M \end{array}$$

Well-Formedness ($\Gamma \vdash V \Leftarrow \text{ok}$)

$$\frac{}{\Gamma \vdash \text{base} \Leftarrow \text{ok}} \qquad \frac{\Gamma \vdash R \Rightarrow \text{base}}{\Gamma \vdash R \Leftarrow \text{ok}} \qquad \frac{\Gamma \vdash V_1 \Leftarrow \text{ok} \qquad \Gamma, x : V_1 \vdash V_2 \Leftarrow \text{ok}}{\Gamma \vdash \Pi x{:}V_1.V_2 \Leftarrow \text{ok}}$$

Checking ($\Gamma \vdash M \Leftarrow V$)

$$\frac{\Gamma, x : V_1 \vdash M \Leftarrow V_2}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}V_1.V_2} \qquad \frac{\Gamma \vdash R \Rightarrow v' \qquad v = v'}{\Gamma \vdash R \Leftarrow v}$$

Synthesis ($\Gamma \vdash R \Rightarrow V$)

$$\frac{x : V \in \Gamma}{\Gamma \vdash x \Rightarrow V} \qquad \frac{\Gamma \vdash R \Rightarrow \Pi x{:}V_1.V_2 \qquad \Gamma \vdash M \Leftarrow V_1 \qquad [M_1/x]V_2 = V_2'}{\Gamma \vdash R\ M \Rightarrow V_2'}$$

Substitution ($[M/x]X = X'$)

$$\frac{[M/x]N = N'}{[M/x](\lambda y.N) = \lambda y.N'} \qquad \frac{[M/x]V_1 = V_1' \qquad [M/x]V_2 = V_2'}{[M/x](\Pi y{:}V_1.V_2) = \Pi y{:}V_1'.V_2'}$$

$$\frac{}{[M/x]\text{base} = \text{base}} \qquad \frac{[M/x]_\beta R = R'}{[M/x]R = R'} \qquad \frac{[M/x]_a R = R'}{[M/x]R = R'}$$

Atomic Substitution ($[M/x]_a R = R$)

$$\frac{}{[M/x]_a y = y} \qquad \frac{[M/x]_a R = R' \qquad [M/x]N = N'}{[M/x]_a(R\ N) = R'\ N'}$$

Reduction ($[M/x]_\beta R = M$)

$$\frac{\dfrac{}{[M/x]_\beta x = M} \qquad [M/x]N = N' \qquad [N'/y]M' = M''}{[M/x]_\beta(R\ N) = M''}$$

Conjecture There is a category whose objects are V such that $\vdash V \Leftarrow \text{ok}$, and whose arrows $V_1 \to V_2$ are terms M such that $x : V_1 \vdash M \Leftarrow V_2$, where composition of arrows is given by substitution.

2007.5.7

It seems that

```
list : base -> base.
bool : base.
true : bool.
false : bool.
```

```
nil : list T.
cons : T -> list T -> list T.
snoc : list T -> T -> list T -> base.
snoc/nil : snoc nil M (cons M nil).
snoc/cons : snoc (cons N S) M (cons N S')
            <- snoc S M S'.

%query 1 1 snoc (cons true (cons false nil)) true X.
```

affords an easy embedding into plain LF like

```
base : type.
obj : base -> type.

list : base -> base.
bool : base.
true : obj bool.
false : obj bool.

nil : obj (list T).
cons : obj T -> obj (list T) -> obj (list T).
snoc : obj (list T) -> obj T -> obj (list T) -> base.
snoc/nil : obj (snoc nil M (cons M nil)).
snoc/cons : obj (snoc (cons N S) M (cons N S'))
            <- obj (snoc S M S').

%query 1 1 obj (snoc (cons true (cons false nil)) true X).
```

2007.5.8

Let terms be defined like degenerate S-expressions by

$$S ::= \langle S, S \rangle \mid \bullet$$

It's not really essential what grammar is taken I think. The type $S \multimap S$ of expressions with exactly one hole is evident. Say a model of computation M is a partial function $S \rightharpoonup \mathbb{B}$. A model M is said to simulate another model N at cost n, written $M \geq_n N$, if there is some $s : S \multimap S$ of 'size' n (by which is meant a count of leaves $\bullet$ in s) such that

$$\forall t. M\ (s \hat{}\ t) = N\ t$$

Lemma 0.2 *If $M_1 \geq_a M_2$ and $M_2 \geq_b M_3$, then $M_1 \geq_{a+b} M_3$.*

Proof Let s witness $M_1 \geq_a M_2$, and s' witness $M_2 \geq_b M_3$. Clearly the composition of linear functions $s \circ s'$ has size $a+b$ by linearity. This is the witness of $M_1 \geq_{a+b} M_3$, for

$$\forall t. M_1\ ((s \circ s')\hat{}\, t) = M_1\ (s\hat{}\,(s'\hat{}\, t)) = M_2\ (s'\hat{}\, t) = M_3\ t$$

■

2007.5.9

Looking back on 2007.3.16: This mechanism could make a fine one-player optimization game of a sort of Railroad Tycoon feel.

Looking back on 2007.3.30: So we have new primitives

$$D_{A,B} : (A \to B) \to (A \to A \multimap B)$$

$$\mathsf{plus}_A : A \,\&\, A \multimap A$$

$$0_A : \top \multimap A$$

Typical rewrites would be

$$D_{A,B_2}(\lambda x.(M\ x)\hat{}\,(N\ x)) =$$

$$\lambda x.\hat{\lambda} u.\mathsf{plus}\langle (M\ x)\hat{}\,((D_{A,B_1}\ N)\ x\hat{}\, u), ((D_{A,B_1 \multimap B_2}\ M)\ x\hat{}\, u)\hat{}\,(N\ x)\rangle$$

(Here $M : A \to (B_1 \multimap B_2)$ and $N : A \to B_1$)

$$D_{A,B_1 \otimes B_2}(\lambda x.(M\ x) \otimes (N\ x)) =$$

$$\lambda x.\hat{\lambda} u.\mathsf{plus}\langle (M\ x) \otimes ((D_{A,B_2}\ N)\ x\hat{}\, u), ((D_{A,B_1}\ M)\ x\hat{}\, u) \otimes (N\ x)\rangle$$

$$D_{A,B_1 \multimap B_2}(\lambda x.\hat{\lambda} v.M\ x\hat{}\, v) =$$
$$\hat{\lambda} v.D_{A,B_2}(\lambda x.M\ x\hat{}\, v)$$

$$D_A(\lambda x.\mathsf{plus}\langle M\ x, N\ x\rangle = \lambda x.\hat{\lambda} u.\mathsf{plus}\langle D_A M\ x\hat{}\, u, D_A N\ x\hat{}\, u\rangle$$

$$D_A(\lambda x.x) = \lambda x.\hat{\lambda} u.u$$
$$D_A(\lambda x.M) = \lambda x.\hat{\lambda} u.0_A$$

$$\mathsf{co}_A : !A \multimap A \multimap !A$$
$$D_{!A}(\lambda x.!(M\ x)) = \lambda x.\hat{\lambda} u.\mathsf{co}\ !(M\ x)\ (M'\ x\hat{}\, u)$$

$$\mathsf{co}_A = \hat{\lambda} u, v.\, \mathbf{let}!y = u\, \mathbf{in}\, \mathcal{D}_{A,!A}(\lambda x.!x)\ y \hat{}\ v\, \mathbf{end}$$

$$D_{!A}(\lambda x.!(M\ x)) = \lambda x.\hat{\lambda} u.\mathcal{D}_{A,!A}(\lambda x.!x)\ (M\ x) \hat{}\ (M'\ x \hat{}\ u)$$

Hm, this looks like I am just using the chain rule, though.

What if I take co as primitive? Instead with type

$$\mathsf{co}_A : A \to A \multimap\ !A$$

Then

$$D_{A,B} = \lambda f : (A \to B).\lambda x : A.\hat{\lambda} u : A.$$

$$\mathbf{let}\, !y = \mathsf{co}_A\ x \hat{}\ u\, \mathbf{in}\, f\ y\, \mathbf{end}$$

The tensor rule becomes

$$\lambda x.\hat{\lambda} u.\, \mathbf{let}\, !y = \mathsf{co}_A\ x \hat{}\ u\, \mathbf{in}(M\ y) \otimes (N\ y))\, \mathbf{end} =$$

$$\lambda x.\hat{\lambda} u.\, \mathbf{let}\, !y = \mathsf{co}_A\ x \hat{}\ u\, \mathbf{in}\, \mathsf{plus}\langle (M\ x) \otimes (N\ y), (M\ y) \otimes (N\ x)\rangle\, \mathbf{end}$$

Here is a possible source of resolution to the Wittgensteinian confusion about rule-following: although there is no *canonical* notion of Kolmogorov complexity that would enable us to apply Occam's razor to determine the "simplest" extension of a set of function values to find the "most natural" output corresponding to a new input, in fact the algorithms employed by actual embodied brains employ related learning algorithms, and so have correlated output hypotheses.

The probability that answers to new questions posed to a pair of agents will coincide is no better than chance *if we suppose those agents to have arbitrary hypothesis spaces.* For better or worse, our brains have an intrinsic notion of simplicity, one perhaps induced by the Kolmogorov measure determined by the physical world. Different brains could have a different measure (and probably they do!) but the signal of similarity is strong enough amidst the noise to make *practical* the assumption that if another human agrees on many data points, she will probably agree on one more.

2007.5.10

Talked to rob simmons about a probabilistic programming language. I had a bit of a struggle understanding what would merit inclusion in a language spec: it seems like you could get a lot of mileage out of just having a data structure for distributions, a function for sampling from them, and a semantics that described distributions over values.

2007.5.11

Suppose there is some underlying security S that takes values over a branching set of possible worlds $\{\epsilon, 0, 1, 00, 01, 10, 11\}$. Suppose further that O has values O_{ij} at the very end, somehow dependent on S's price.

Can we compute O's value at world i by replication? S will be worth S_{i0} or S_{i1} at the next step, and O will be worth O_{i0} or O_{i1}. We want p_S and p_B so that

$$\begin{bmatrix} S_{i0} & 1 \\ S_{i1} & 1 \end{bmatrix} \begin{bmatrix} p_S \\ p_B \end{bmatrix} = \begin{bmatrix} O_{i0} \\ O_{i1} \end{bmatrix}$$

so

$$\begin{bmatrix} p_S \\ p_B \end{bmatrix} = \begin{bmatrix} S_{i0} & 1 \\ S_{i1} & 1 \end{bmatrix}^{-1} \begin{bmatrix} O_{i0} \\ O_{i1} \end{bmatrix}$$

And so the actual value O_i is

$$\begin{bmatrix} S_i & 1 \end{bmatrix} \begin{bmatrix} p_S \\ p_B \end{bmatrix} = \begin{bmatrix} S_i & 1 \end{bmatrix} \begin{bmatrix} S_{i0} & 1 \\ S_{i1} & 1 \end{bmatrix}^{-1} \begin{bmatrix} O_{i0} \\ O_{i1} \end{bmatrix}$$

$$= \begin{bmatrix} S_i & 1 \end{bmatrix} \frac{1}{S_{i0} - S_{i1}} \begin{bmatrix} 1 & -1 \\ -S_{i1} & S_{i0} \end{bmatrix} \begin{bmatrix} O_{i0} \\ O_{i1} \end{bmatrix}$$

$$= \begin{bmatrix} S_i & 1 \end{bmatrix} \frac{1}{S_{i0} - S_{i1}} \begin{bmatrix} O_{i0} - O_{i1} \\ O_{i1}S_{i0} - O_{i0}S_{i1} \end{bmatrix}$$

$$= \frac{S_i(O_{i0} - O_{i1}) + O_{i1}S_{i0} - O_{i0}S_{i1}}{S_{i0} - S_{i1}}$$

Now for the case that $S_{i0} = S_i + h$ and $S_{i1} = S_i - h$:

$$= \frac{S_i(O_{i0} - O_{i1}) + O_{i1}(S_i + h) - O_{i0}(S_i - h)}{2h}$$

$$= \frac{O_{i1}h + O_{i0}h}{2h}$$

$$= \frac{O_{i1} + O_{i0}}{2}$$

This seems really counterintuitive, because the value of a call option would then increase without bound as the approximation time increment goes to zero.

Wait, no: the number of possible worlds in which the stock price goes off to infinity still gets dwarfed by the worlds where it doesn't. The model simply seems to impose analysis in terms of a normal distribution. You just want to take a dot product of some normal centered at the current

underlying price against the derivative's payoff function. How does one figure out the right variance for the normal, though?

2007.5.12

Working on revising that paper about non-type-indexed hereditary substitution. Turns out I neglected to be careful about a few basic lemmas involving the interaction of $\sqsubseteq$ and $\cup$.

2007.5.13

The induction measure for untyped hereditary substitution associativity is significantly subtle than I expected.

2007.5.14

The stochastic integral

$$\int f(t)\, dW_t$$

is kind of like a dot product of f against the derivative of the Weiner process — which is peculiar, since the latter a.c. doesn't exist.

2007.5.15

1. Labelled linear unification might be best accomplished by explicit typing constraints mixed in with equality constraints. Something like

   $$\exists X : \Gamma \vdash A[p].P$$

   might be translated to

   $$\exists X : \Gamma \vdash A^{-}.P \wedge (\Gamma \vdash X \Rightarrow A[p])$$

   Then you might have transitions like (assuming $c : B \in \Sigma$)

   $$(\Gamma \vdash R \Leftarrow A[p]) \mapsto (\Gamma \vdash R \Rightarrow A[p])$$

   $$(\Gamma \vdash c \cdot S \Rightarrow A[p]) \mapsto (\Gamma \vdash S : B[\epsilon] > A[p])$$

   $$(\Gamma \vdash () : A[p] > A'[p']) \mapsto (A' \doteq A) \wedge (p \doteq p')$$

   and

   $$(\Gamma \vdash (M; S) : \Pi x{:}A.B[p] > C[q]) \mapsto$$

   $$(\Gamma \vdash M \Leftarrow A[\epsilon]) \wedge (\Gamma \vdash S : [M/x]^{A} B[p] > C[q])$$

   and

   $$(\Gamma \vdash S : \forall \alpha.B[p] > C[q]) \mapsto$$

   $$\exists \beta : (\Gamma \vdash \mathsf{w}).(\Gamma \vdash S[[\beta[\mathsf{id}_{\Gamma}]/\alpha]p] : B > C[q])$$

2. Consider the LF self-embedding

$$\Sigma \mathrel{+}= (\text{base} : \text{type}, \text{obj} : \text{base} \to \text{type})$$

$$\text{type} \mapsto \text{base}$$

$$a \cdot S \mapsto \text{obj}(a \cdot S)$$

   There are valid *LF* expressions *not* in the image of this translation, such as base $\to$ base, and, if we consider *LLF*, things such as $A \multimap$ base. I especially wonder whether the latter could make sense.

3. Perhaps the argument for mere termination of substitution should be reworked to involve set notation to match that necessary for the substitution associativity argument.

Do I need to be careful about contextual status during typechecking even?

Harland & Pym.

Non-well-moded examples?

2007.5.16

Degenerate typing in spineless form:

$$(x \in R\ N)^j_r = (x \in R)^{tp(N) \to j}_r \cup (x \in N)_n$$

$$(x \in x)^j_r = \{t\} \qquad (x \in y)^j_r = \emptyset$$

$$(x \in R)_n = (x \in R)^o_r$$

$$(x \in \lambda y.N)_n = (x \in N)_n$$

$$tp(R) = o$$

$$tp(\lambda x.N) = (x \in N) \to tp(N)$$

2007.5.17

Kaustuv was trying to explain to me a focussing system where you focus on multiple things at once.

2007.5.18

Read some lexical semantics notes linked to by Noah Smith that Rob Simmons told me about. Awfully frustrating to see several vaguely FOLish schemata described as possible targets for representing propositional utterances and queries, but to not have any sense of what would make any of them more suitable than the others.

2007.5.19

Meeting with Kaustuv and Frank: Kaustuv described a system he was working on that combined a hybrid-style temporal logic with linear logic, for expressing systems that changed state consumptively but also with specific delays. I found it troublingly difficult to see how to squeeze in the notion that a reaction *will* take place if it can, but there is an intuitionistically one-sided way of looking at it that still works: a reaction *can* take place with a certain timing (where all delays act like lower bounds) iff the system proves some corresponding sequent.

Another interesting thing was the fact that expressing a temporal box modality begged for the same trick that tom7 needed to get accessibility facts *intrinsically* mobile.

2007.5.20

A thought about polymorphism in LF. With general predicative polymorphism, subordination is instantly shot to hell, even with relatively well-meaning types. Like if I say

```
list : type -> type.
nil : list T.
cons : T -> list T -> list T.
```

Then I can instantiate `T` with any function type $A \to B$ and A is subordinate to B. Calculating subordination with base-type polymorphism might still be difficult — a naïve point of view might have it that every type subordinates the coalesced type *operator* `list`, whereas a more clever definition might notice that only `list T` and `T` subordinate the individual type `list T` — but it seems sane, at least.

Adapting regular unification to contextual unification seems easy for ACU — just split equations along the different parameters, and set to ε any variables that aren't allowed to depend — but I don't know what to do for *AU*.

2007.5.21

The boolean variable approach of Harland and Pym looks a lot like the reduction of contextual ACU unification to solving equations over $\mathbb{N}$ (or the finite subset of it, 2) which makes be pessimistic about it extending well to ordered logic.

2007.5.22

Why is $\otimes$ left asynchronous but not obviously a left adjoint to anything? I suppose 'half' of it is left adjoint to $\multimap$, but this answer doesn't seem

satisfying somehow.

2007.5.23

If I wanted to imagine a collection of abstracted, modifiable names existing at the beginning of, say, a twelf file, I would want each one of them to be defined exactly one place. However, each would be *used* many times in the bodies of definitions or declarations or what-have-you, and certainly possibly zero times in a particular declaration, so a straightforward use of linearity would not work.

It seems the zero-ary (but not proof-irrelevant) arrow is exactly what's needed: a condec would take a name linearly, but take the defining (or classifying) right-hand side "zeroarily".

2007.5.24

Sitting in on a meeting with some NLP people. It strikes me again how incredibly difficult natural languages are to deal with — but I keep hoping that by solving harder (more general) problems the specific case of 'understanding' parsed sentences might be made easer. For it's not merely declarative things that we learn, and not merely the semantic maps in language, but we also *learn* to parse, and *learn* to distinguish nouns from verbs, and *learn* even to segment words out of continuous speech. Since all of these tasks *are* learned, the linguistic behaviors we habitually engage in are sloppy and noisy: and so they are hard for carefully engineered systems to treat.

2007.5.25

If descending into children of a tree node preferentially chooses the most recently accessed child, then you get a sort of local inverse relationship between "most recent child of" and "parent of". Clearly however, they can't be real inverses if you account for all of the state involved, because arriving at a child clobbers the former information of the child accessed before that.

2007.5.26

Incremental bidirectional preorder traversal is not that hard to implement, but it still feels ugly.

2007.5.27

In linear logic, have a multiconclusion calculus with the right side interpreted tensorially. There is another judgment besides ordinary linear truth which admits a superscript which signifies tensorial 'number of copies'. Structural rules:

$$\frac{\Gamma, A^m, A^n \vdash \Delta}{\Gamma, A^{m+n} \vdash \Delta} \qquad \frac{\Gamma \vdash A^m, A^n, \Delta}{\Gamma \vdash A^{m+n}, \Delta}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A^1 \vdash \Delta} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A^1}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A^0 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A^0, \Delta}$$

$$\frac{\Gamma \vdash \Delta \qquad \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \qquad \frac{}{\cdot \vdash \cdot}$$

Connectives:

$$\frac{\Gamma \vdash [a/x]A}{\Gamma \vdash \forall x.A} \qquad \frac{\Gamma, [n/x]A \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}$$

$$\frac{\Gamma \vdash A^n}{\Gamma \vdash \otimes^n A} \qquad \frac{\Gamma, A^n \vdash \Delta}{\Gamma, \otimes^n A \vdash \Delta}$$

Question: how much like $!A$ is $\forall x. \otimes^x A$? Is this the 'other polarity' version of !? (asynch-then-synch instead of synch-then-asynch)

Better yet; the only judgments (on the left or right) are A^n where

$$n ::= 1 \mid a \mid x$$

where a is a parameter. The judgment A^1 is identified with A. We write ν for a sequence of ns. $A^{n_1,\ldots,n_N}$ means $A^{n_1}, \ldots, A^{n_N}$. Structural rules:

$$\frac{\Gamma \vdash \Delta \qquad \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \qquad \frac{}{\cdot \vdash \cdot}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma^a \vdash \Delta^a}$$

Connectives:

$$\frac{\Gamma \vdash [a/x]A}{\Gamma \vdash \forall x.A} \qquad \frac{\Gamma, [\nu/x]A \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}$$

$$\frac{\Gamma \vdash A^\nu, \Delta}{\Gamma \vdash \otimes^\nu A, \Delta} \qquad \frac{\Gamma, A^\nu \vdash \Delta}{\Gamma, \otimes^\nu A \vdash \Delta}$$

Abbreviate $!A \equiv \forall x. \otimes^x A$. It seems I (correctly) still can't prove

$$\frac{\dfrac{A \multimap B, A^a \vdash B^a}{A \multimap B, !A \vdash B^a}}{A \multimap B, !A \vdash !B}$$

How about $\forall x.(A \otimes B) \equiv (\forall x.A) \otimes B$? No,

$$\forall x.(A(x) \otimes B) \vdash (\forall x.A(x)) \otimes B$$

fails.

Dang, I can't seem to prove

$$\dfrac{\dfrac{\dfrac{A \vdash !A}{A^a \vdash (!A)^a}}{!A \vdash (!A)^a}}{!A \vdash !!A}$$

Need multiplication or something. Like:

$$\dfrac{\Gamma \vdash A^{\nu n}, \Delta}{\Gamma \vdash (\otimes^\nu A)^n, \Delta} \qquad \dfrac{\Gamma, A^{\nu n} \vdash \Delta}{\Gamma, (\otimes^\nu A)^n \vdash \Delta}$$

Where ns are now multiplicative lists of parameters. I would then get

$$\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A}{A^{ab} \vdash A^{ab}}}{!A \vdash A^{ab}}}{!A \vdash (!A)^a}}{!A \vdash !!A}$$

2007.5.28

To prove cut-freeness for the system from yesterday I seem to need to generalize to the admissibility of something like

$$\dfrac{\Gamma \vdash \Psi \qquad \Psi \mapsto_\otimes \Psi' \qquad \Delta, \Psi' \vdash \Omega}{\Gamma, \Delta \vdash \Omega}$$

Where $\mapsto_\otimes$ indicates that $\Psi \vdash \Psi'$ can be proved by introduction rules for $\otimes$.

The case analysis here would grind away on $\Gamma \vdash \Psi$ until it uncovered an instance of a synchronous 'mix' rule like

$$\dfrac{\Gamma_i \vdash A_i^{n_i} \quad (\forall i)}{\Gamma_1, \ldots, \Gamma_N \vdash A_1^{n_1}, \ldots, A_N^{n_N}}$$

to match the synchronous decomposition of $\otimes$ that has already taken place on the left in $\Delta, \Psi' \vdash \Omega$ splitting up Ψ'.

What's going on with $\forall$s in my thesis's logic seems to be an indexed version of the phenomenon of something like

$$a : o \to \text{type}$$

$$k : o$$

$$s_1, s_2, s_3 \sqsubset a :: o \to \text{type}$$

$$c : s_1\ k \wedge s_2\ k \wedge s_3\ k$$

One might think as the limit as being something like

$$a : o \to \text{type}$$

$$k : o$$

$$s \sqsubset a :: \Pi i{:}w.o \to \text{type}$$

$$c : \forall i.s\ i\ k$$

I feel this means the right interpretation of kind-level refinements is that

$$\frac{\Gamma \vdash A : \text{type}}{\Gamma \vdash \mathbf{ref}(A) : \text{kind}}$$

And the usual

$$s \sqsubset a :: \Pi x_1{::}S_1 \ldots \Pi x_n{::}S_n.\text{type}$$

should be

$$s : \Pi x_1{::}S_1 \ldots \Pi x_n{::}S_n.\mathbf{ref}(a \cdot (x_1; \cdots; x_n))$$

But then $\leq$ might get undecidable depending on how you do things. I think this interpretation makes the translation clearer, though.

2007.5.29

James Cheney made an LJ post about the combinatorics problem of counting closed lambda terms. I wonder what the generating function is?

Say c_{mn} is the number of lambda terms of depth at most m over n free variables. We know:

$$c_{0n} = 0$$

$$c_{(m+1)n} = n + c_{mn} * c_{mn} + c_{m(n+1)}$$

So $f(x, y) =$

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} x^m y^n c_{mn} = \sum_{m=1}^{\infty}\sum_{n=0}^{\infty}(n + c^2_{(m-1)n} + c_{(m-1)(n+1)})x^m y^n$$

$$= \left(\sum_{m=1}^{\infty} \sum_{n=0}^{\infty} n x^m y^n \right) + \left(\sum_{m=1}^{\infty} \sum_{n=0}^{\infty} c^2_{(m-1)n} x^m y^n \right) + \cdots$$

urg, I'll probably never get rid of that c^2.

How about lambda terms of *size* m over n free variables?

$$c_{0n} = 0$$

$$c_{(m+1)n} = n + \left(\sum_{i=0}^{m} c_{in} * c_{(m-i)n} \right) + c_{m(n+1)}$$

Here $f(x, y) =$

$$\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} x^m y^n c_{mn} = \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} (n + \left(\sum_{i=0}^{m} c_{in} c_{(m-i)n} \right) + c_{(m-1)(n+1)}) x^m y^n$$

$$= A + B + C$$

where

$$A = \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} n x^m y^n$$

$$= y \left(\left(\sum_{n=1}^{\infty} -n y^{n-1} \right) + \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} n x^m y^{n-1} \right)$$

$$B = ugh$$

2007.5.30

It seems that something like parsing can be done just in terms of displayed traversal of tree nodes and possible insertion of auxiliary 'parenthesis' nodes.

2007.5.31

I just remembered why e^x actually converges:

$$\left(1 + \frac{x}{n}\right)^n = \sum_{i=0}^{n} \left(\frac{x}{n}\right)^i \binom{n}{i}$$

$$= \sum_{i=0}^{n} \left(\frac{x}{n}\right)^i \left(\frac{n \cdots (n - i + 1)}{i!} \right)$$

$$\leq \sum_{i=0}^{n} \left(\frac{x}{n}\right)^i \left(\frac{n^i}{i!} \right)$$

$$= \sum_{i=0}^{n} \frac{x^i}{i!}$$

So

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n \leq \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

And we can see the latter thing converges absolutely for any x because as soon as $i > 2\|x\|$ each successive term has norm less than half of the previous one. Formally,

$$\left\| \sum_{i=0}^{\infty} \frac{x^i}{i!} \right\| = \left\| \left(\sum_{i=0}^{\lceil 2\|x\| \rceil} \frac{x^i}{i!} \right) + \sum_{i=\lceil 2\|x\| \rceil}^{\infty} \frac{x^i}{i!} \right\|$$

$$\leq \left\| \left(\sum_{i=0}^{\lceil 2\|x\| \rceil} \frac{x^i}{i!} \right) + \lceil 2\|x\| \rceil \sum_{i=0}^{\infty} 2^{-i} \right\|$$

$$= \left\| \left(\sum_{i=0}^{\lceil 2\|x\| \rceil} \frac{x^i}{i!} \right) + 2\lceil 2\|x\| \rceil \right\| < \infty$$

$$\leq \sum_{i=0}^{n} \left(\frac{x^i}{i!} \right) \left(\frac{n \cdots (n-i+1)}{n^i} \right)$$

2007.6.1

Remembering this type derivative stuff. Suppose we want to figure out the derivative of a μ. Say $S(\beta) = \mu\alpha.T(\alpha, \beta)$. Compute

$S'(\beta) = \frac{d}{d\beta} S(\beta)$
$= \frac{d}{d\beta} \mu\alpha.T(\alpha, \beta)$
$= \frac{d}{d\beta} [\mu\alpha.T(\alpha, \beta)/\alpha] T(\alpha, \beta)$
$= \frac{d}{d\beta} T(\mu\alpha.T(\alpha, \beta), \beta)$
$= \frac{d}{d\beta} T(S(\beta), \beta)$
$= S'(\beta) T_1(S(\beta), \beta) + T_2(S(\beta), \beta)$
$\therefore S'(\beta) = S'(\beta) * T_1(S(\beta), \beta) + T_2(S(\beta), \beta)$
$S'(\beta) = \mu\alpha.(\alpha * T_1(S(\beta), \beta) + T_2(S(\beta), \beta))$
$S'(\beta) = T_2(S(\beta), \beta) * (T_1(S(\beta), \beta)\ \text{list})$

2007.6.2

A story, expanded from a dream:

It started with the elevator. I pushed eight, but it passed it on the way up and went to nine. The doors were always locked on nine, and the nine-button never worked. Surely it was where spare pipes were kept, spare plugs and miscellaneous tools and control panels for the building. But no: (as I stepped out, the two ladies beamed shame at me; duchesses condemning a violation of etiquette) I found only a corridor with an uneven floor, not merely cobblestoned but of gross irregularity, rolling like seasick waves, vertiginous and unstable yet maddeningly unmoving.

The windows were narrow, and twice my height. Through them I saw a wall of graffiti, a policeman conceiving a plan to erase all of it in one afternoon. The tags' letters I thought beautiful, sunk in shadows both painted and cast.

I found the stairs, returned to the eighth floor. I was late for class.

Seated, doors open to my left now (noisily, ominously) shut, I found two sheets of paper each lined on one side. A sealed envelope of instructions. A girl next to me, (blonde, not my type) her attention only on the front of the room. We were told it was time to open the envelopes.

A role-playing exercise. The instructions contradicted themselves frequently, made us feel (they must have intended) inhabitants of an uncomfortable future, where cold reasoning was socially acceptable only in doses of under 50 milligrams at a time, ask a doctor for children under twelve, get a refill on your prescription. Warmth was equally discouraged. No kissing in public, (bus-water on bus-water) no hand-holding, no short pants, fill both sides of both sheets of paper with writing on the provided theme, ignore the instructions, ignore the instructions, ignore the theme, ignore the provided story, do not analyze it, do not answer formal questions about it in the argumentative structure with which you are by now familiar.

What a shallow mindfuck — I thought it cheap, juvenile. Still, I won the game, such as it was, the winning move being, first, a note passed to my right, the paper lined on one side. I confess I found their (partly) staged fury satisfying, but not more so than the essential and internal feeling of transgression itself. I had broken one rule, and followed another, the latter being silent and (being, some say, our only device against mortality) immortal. From what I heard the following day, she enjoyed it, too, but still she never spoke directly to me during classes, or in the hallways, or under the wide eaves (the only shaded place at midday) spreading from the top of the ninth floor.

It's been fourteen years. This morning, her letter arrived.

2007.6.3

Here's an attempt to encode Yablo's paradox in hybrid logic. Let $X = \forall t.A@t \Leftrightarrow \forall n \geq 1.\neg(A@(t+n))$. I aim to prove $\neg X$.

Suppose $A@n$. Then by X, we have $\neg A@(n+1)$. But we can also specialize $\forall n.\neg(A@(t+n))$ to $\forall n.\neg(A@(t+n+1))$, but X tells us this is equivalent to $A@(n+1)$. So we have a contradition. Therefore $\neg A@n$. But n is arbitrary, so we have shown $\forall t.\neg A@t$. But this is $A@0$. We have a contradiction from only the assumption X.

* * * * * *

I think I can do this with the monoid domain too. Consider $\Gamma = \alpha : \mathsf{w}, f : \tau_f, g : \tau_g$ where

$$\tau_f = \forall t.(A@t \to \forall p.(A@(t * p * \alpha) \to C))$$

and

$$\tau_g = \forall t.((\forall p.(A@(t * p * \alpha) \to C)) \to A@t)$$

Imagine some $\beta : \mathsf{w}$, and $x : A@\beta$. Then we can see $f\ x : \forall p.(A@(\beta * p * \alpha) \to C)$, and so also (plugging in ϵ for p) we get $f\ x : A@(\beta * \alpha) \to C$. Now we're going to try to use g to try to make something to plug into this function. Instantiate $t = \beta * \alpha$ to get

$$g : (\forall p.(A@(\beta * p * \alpha * \alpha) \to C)) \to A@(\beta * \alpha)$$

We should also see that

$$f\ x : \forall p.(A@(\beta * p * \alpha * \alpha) \to C)$$

So $g\ (f\ x) : A@(\beta * \alpha)$ hence also $f\ x\ (g\ (f\ x)) : C$. Abstracting over what we've build up,

$$\lambda x.f\ x\ (g\ (f\ x)) : \forall \beta.(A@\beta \to C) \qquad (*)$$

so too

$$\lambda x.f\ x\ (g\ (f\ x)) : \forall \beta.(A@(\beta * \alpha) \to C)$$

which fits into g as long as we choose $t = \epsilon$.

$$g\ (\lambda x.f\ x\ (g\ (f\ x))) : A@\epsilon$$

but I can plug this into x in $(*)$, obtaining a closed term of type C, namely

$$f\ (g\ (\lambda x.f\ x\ (g\ (f\ x))))\ (g\ (f\ (g\ (\lambda x.f\ x\ (g\ (f\ x))))))$$

To η-expand this:

$$f$$
$$(g\ (\lambda x.f\ x\ (g\ (\lambda y.f\ x\ y))))$$
$$(g\ (\lambda z.f\ (g\ (\lambda x.f\ x\ (g\ (\lambda y.f\ x\ y))))\ z))$$

Note that f and g look kind of like *app* and *lam*. If we interpret them that way, we get the object language term

$$(\lambda x.x\ (\lambda y.x\ y))$$

$$(\lambda z.(\lambda x.x\ (\lambda y.x\ y))\ z)$$

Defining $\Omega = \lambda x.x\ x$, this is the usual $\Omega\ \Omega$ up to η-expansion.

Listening to talks at a formal epistemology workshop ('FEW 2007'), which is going on here at CMU.

The 'equal epistemic competence' assumption in Tomoji Shogenji's talk is interesting — they seem to be saying there is one (scalar!) notion of competence, erasing the necessity to talk about some isomorphism between the epistemic model of one agent and another. Kevin Kelly asked a question about what becomes of an ultra-Bayesian point of view where one has simply a huge model of other agents' cognition as parts of the world, and in that case, there is a definite nontrivial isomorphism between *my* huge model, and yours, for you are reduced to a ball of priors and conditional probabilities in my model, and *I* am so reduced in yours.

There are some agent-indexical propositions ('my feelings are normatively important', 'my perceptions are not illusory') that survive transport across these isomorphisms in an interesting and non-identical way!

* * * * * *

One thing that keeps striking me about this notion of *averaging* the opinions of experts is a worry about scaling properties. There was some axiom in Alon Altman's work on axiomatizations of ranking systems, I think, that cloning agents shouldn't affect ranking.

2007.6.4

Deepak told me some neat things about $\bigcirc$, namely that $\bigcirc A$ is equivalent to $(A \multimap p) \multimap p$ for some fresh atom p. I'm surprised this isn't in the Frank/Evan/Kaustuv paper on JILL.

2007.6.5

Neel considered the following principle mentioning proof irrelevance, which seems somewhat classical but weaker than Markov's principle:

$$(\forall x.P(x) \vee \neg P(x)) \Rightarrow [\exists x.P(x)] \Rightarrow \exists x.P(x)$$

2007.6.6

Have mostly succeeded in construing infix, prefix, postfix, and n-ary circumfix operators (e.g. list literals) as all special cases of the same thing, also including constructs like λ-binding, and if-then-else.

2007.6.7

Turns out getting a relatively uniform method for unparsing is still tricky. I may resort to something more concrete, i.e. a more flexible macro-ish system with output in a typical formatting combinator language.

2007.6.8

Using curses, `erase` is much better than `clear`; the latter actually repaints everything, whereas the former does what is expected of curses, namely conservative repainting of only what is required.

2007.6.9

I should ask neel about his thoughts on arranging intuitionistic proofs to have explicit comonadic ϵliminations and δuplications of 'resources' — I think something like that would be necessary to compile uses of zippers into imperative functions that statefully update data in the same way as some given functional zipper-transformer. Moreover this would make a nice application for linear metareasoning if my thesis were closer to done.

Perhaps the right way to think about the irrelevance in world-choices in the labelled system is to push *all* the irrelevance off to the side; a well-typed term in the labelled calculus is first well-typed as an ordinary term, and then there must be a separate (irrelevant) proof that it belongs to the suitable refinement.

Does dependency thwart this? I can't tell right away; worlds don't appear in the 'proper' terms, so perhaps not.

2007.6.10

Maybe substitution inversion is the right locus for kicking off label unification also.

2007.6.11

Like, say $u :: \Gamma_u \vdash A[p]$ is in Δ somewhere. Faced with

$$u[\sigma] = M$$

we would try to figure that

$$u \leftarrow M[\sigma^{-1}]$$

but we'd need to check that $M[\sigma^{-1}]$ uses resources p. Now $\Gamma \vdash \sigma : \Gamma_u$ for the ambient context Γ, right, so $\Gamma_u \vdash \sigma^{-1} : \Gamma$. Meanwhile we check that $\Gamma \vdash M : B[q]$ and if we hit everything with σ^{-1} we get something like

$$\Gamma_u \vdash M[\sigma^{-1}] : B[\sigma^{-1}][q[\sigma^{-1}]]$$

2007.6.12

Imagine U is a kind of some constructors that only exist as variables, and that $elm(u)$ is a type whenever $u : U$. There are some other things called 'paths' π that are classified by expressions $u \rightsquigarrow v$.

We can only type*check* pairs consisting of an atomic thing of type elm together with a coercion path, which is meant to be interpreted as proof-irrelevant.

$$\frac{\Gamma \vdash R \Rightarrow elm(u) \qquad \Gamma \vdash \pi : u \rightsquigarrow v}{\langle R, \pi \rangle \Leftarrow elm(v)}$$

Coercion paths are built up like

$$\frac{x : u \rightsquigarrow v \in \Gamma}{\Gamma \vdash x : u \rightsquigarrow v} \qquad \frac{}{\Gamma \vdash id : u \rightsquigarrow u} \qquad \frac{\Gamma \vdash \pi : u \rightsquigarrow v \qquad \Gamma \vdash \pi' : v \rightsquigarrow w}{\Gamma \vdash \pi; \pi' : u \rightsquigarrow w}$$

2007.6.13

Or perhaps:

$$\frac{\Gamma \vdash R \Rightarrow a \cdot S \qquad \Gamma \vdash P : S \rightsquigarrow S' : A > \text{type}}{\langle R, P \rangle \Leftarrow a \cdot S'}$$

with coercions

$$\frac{x : u \rightsquigarrow v : A \in \Gamma}{\Gamma \vdash x : u \rightsquigarrow v : A} \qquad \frac{\Gamma \vdash \pi : u \rightsquigarrow v : A \qquad \Gamma \vdash \pi' : v \rightsquigarrow w : A}{\Gamma \vdash \pi; \pi' : u \rightsquigarrow w : A}$$

$$\frac{}{\Gamma \vdash () : () \rightsquigarrow () : \text{type} > \text{type}}$$

$$\frac{\Gamma \vdash \pi : M \rightsquigarrow M' : A \qquad \Gamma \vdash P : S \rightsquigarrow S' : [M'/x]^A B > \text{type}}{\Gamma \vdash (\pi; P) : (M; S) \rightsquigarrow (M'; S') : \Pi x{:}A.B > \text{type}}$$

I would hope for the lemma

Lemma 0.3 *If* $\Gamma \vdash M \rightsquigarrow M' : A$ *and* $\Gamma \vdash N : [M/x]^A B$, *then* $\Gamma \vdash N : [M'/x]^A B$.

This would justify the asymmetric substitution of M' in the spine cons rule immediately above. However, since x might occur negatively, I might need $\rightsquigarrow$ to actually be equality.

2007.6.14

Alice and Bob agree that the sun has always risen in the past, and that it is likely to rise again tomorrow morning. But the next morning arrives, and Bob is alarmed to find that the sun does not rise, but instead a strange, yellowish orb that he decides to call a 'smun'. Alice, however, says that this object *is* the sun, and it has risen as usual.

Obvious moral: inductive and abductive reasoning depends on our notions of identity of objects and phenomena across time.

I am suspicious of the possibility of monosemy, that a word could in any sense have *one* meaning, but we would certainly like to constrain our use of language to words (or morphemes, or sentences) that have *clear* and *stable* denotations.

A big problem is assessing what I am trying to get at by saying 'stable'. Bob claims that his definition of 'sun' is stable, and that the yellowish orb in the sky is not included in it; and yet Alice claims her definition of 'sun' is stable too.

This whole business is scarcely different from the classical 'grue' argument, maybe not at all. I thought at first that it was, but I'm increasingly doubting that I can find any difference.

It is a profound and *empirical* fact that our notions of stability coincide so often!

2007.6.15

Another try:

$$\pi ::= \lambda x.\pi \mid u \cdot P \mid \pi_1; \pi_2 \mid \widetilde{\pi}$$

$$P ::= () \mid (\pi; P)$$

$$R ::= H \cdot S$$

$$N :: \lambda x.N \mid R \bullet P$$

$$\frac{a : K \in \Sigma \qquad \Gamma \vdash R \Rightarrow a \cdot S \qquad \Gamma \vdash P : S \sim S' : K > \text{type}}{R \bullet P \Leftarrow a \cdot S'}$$

with coercions typed by

$$\frac{\Gamma \vdash \pi : M \sim M' : A \qquad \Gamma \vdash P : S \sim S' : [M/x]^A V > W}{\Gamma \vdash (\pi; P) : (M; S) \sim (M'; S') : \Pi x{:}A.V > W}$$

$$\frac{}{\Gamma \vdash () : () \sim () : \text{type} > \text{type}}$$

$$\frac{\Gamma \vdash \pi : M \sim N : A}{\Gamma \vdash \widetilde{\pi} : N \sim M : A} \qquad \frac{\Gamma \vdash \pi : M \sim N : A \qquad \Gamma \vdash \pi' : N \sim P : A}{\Gamma \vdash \pi; \pi' : M \sim P : A}$$

$$\frac{u : M \sim N : A \in \Gamma \qquad \Gamma \vdash P : S \sim S' : A > C}{\Gamma \vdash u \cdot P : [M|S]^A \sim [N|S']^A : C}$$

$$\frac{\Gamma \vdash H \Rightarrow A \qquad \Gamma \vdash P : S \sim S' : A > C}{\Gamma \vdash H \cdot P : H \cdot S \sim H \cdot S' : C}$$

$$\frac{\Gamma, x : A \vdash \pi : M \sim N : B}{\Gamma \vdash \lambda x.\pi : \lambda x.M \sim \lambda x.N : \Pi x{:}A.B}$$

But I shouldn't really bother about canonical forms at the proof level. Something more like:

$$\pi ::= \lambda x.\pi \mid u \mid H \mid \pi_1\ \pi_2 \mid \pi_1; \pi_2 \mid \widetilde{\pi} \mid id$$

$$\frac{u : M \sim N : A \in \Gamma}{\Gamma \vdash u : M \sim N : A} \qquad \frac{\Gamma \vdash H \Rightarrow A}{\Gamma \vdash H : H \sim H : A}$$

$$\frac{\Gamma \vdash \pi_1 : \lambda x.M_1 \sim \lambda x.N_1 : \Pi x{:}A.B \qquad \pi_2 : M_2 \sim N_2 : A}{\Gamma \vdash \pi_1\ \pi_2 : [M_2/x]^A M_1 \sim [N_2/x]^A N_1 : [M_1/x]^A B}$$

$$\frac{}{\Gamma \vdash id : M \sim M : A}$$

2007.6.16

Consider the relationship between 'is' and 'seems'. The latter emphasizes that some attribution is *not certain*. Imagine a (meditative) verb X that emphasizes the attribution is *not important*. E.g. "I X upset" means "I am upset, but it does not matter; this is merely a transient state which does not merit lasting attention, and which I am capable of ignoring".

2007.6.17

I think I have some better understanding of why the unification problems that arise from pure LLF (or OLF) are easily decidable.

Let $C(\Gamma; p)$ be the set of unification problems that can arise from a query like $\Gamma \vdash M \Leftarrow p$ and $R(\Gamma; p)$ the set of r that can arise from $\Gamma \vdash S : A[p] > C[r]$, and $S(\Gamma; p)$ the set of unification problems that arise from same.

Based on

$$\frac{\Gamma, \alpha : \mathsf{w}, x : A@\alpha \vdash M \Leftarrow B[p * a]}{\Gamma \vdash \lambda x.M \Leftarrow A \multimap B}$$

We can see $C(\Gamma, \alpha, x; p * \alpha) \subseteq C(\Gamma; p)$. Based on

$$\frac{\Gamma, x : A \vdash M \Leftarrow B[p]}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}A.B[p]}$$

We get only the trivial $C(\Gamma, x; p) \subseteq C(\Gamma; p)$.

From

$$\frac{(\alpha : \mathsf{w}, x : A@\alpha) \in \Gamma \qquad \Gamma \vdash S : A[\alpha] > C[r] \qquad r = p}{\Gamma \vdash x \cdot S \Leftarrow C[p]}$$

we see that $\{e \wedge r \doteq p \,|\, e \in S(\Gamma; \alpha), r \in R(\Gamma; \alpha)\} \subseteq C(\Gamma; p)$. From

$$\frac{x : A \in \Gamma \qquad \Gamma \vdash S : A[\epsilon] > C[q] \qquad r = q}{\Gamma \vdash x \cdot S \Leftarrow C[r]}$$

we see that $\{e \wedge r \doteq p \,|\, e \in S(\Gamma; \epsilon), r \in R(\Gamma; \epsilon)\} \subseteq C(\Gamma; p)$.

From the & and $\top$ introduction rules we see $C(\Gamma; p)$ is closed under conjunction and contains the trivial unification problem $\top$. Consider now the $\multimap$ elimination rule:

$$\frac{\Gamma \vdash M : A[q] \qquad \Gamma \vdash S : B[p * q] > C[r]}{\Gamma \vdash (M; S) : A \multimap B[p] > C[r]}$$

Therefore we make up a new evar Q and see that

$$\{e_1 \wedge e_2 \,|\, e_1 \in C(\Gamma, Q; Q), e_2 \in S(\Gamma, Q; p * Q)\} \subseteq S(\Gamma; p)$$

and $R(\Gamma, Q; p{*}Q) \subseteq R(\Gamma; p)$. From the unrestricted application we get only

$$\frac{\Gamma \vdash M : A[\epsilon] \qquad \Gamma \vdash S : B[p] > C[r]}{\Gamma \vdash (M; S) : \Pi x{:}A.B[p] > C[r]}$$

and see only trivial things like $S(\Gamma; p) \subseteq S(\Gamma; p)$. At the nil case

$$\frac{}{\Gamma \vdash () : A[p] > A[p]}$$

we get the base case $p \in R(\Gamma; p)$ and $\top \in S(\Gamma; p)$.

2007.6.20

I need to make up my mind whether world terms are present in modal substitutions.

2007.6.24

It seems not at all feasible to maintain well-labelledness as an invariant during unification, because of the spine cons case — there we don't know

what world to plug into the tail, even though in principle there is an answer that would work if we knew what equation to demand on the labels.

2007.6.25

The idea of a folding editor leaving actual comments (i.e. fold marks) *in* the file seems peculiar, but maybe it's unavoidable.

2007.6.26

I want to say: all a word's meaning is, is its use. But what I would better say is: a word's use is at least a somewhat clear notion. Let us begin by thinking about that, and avoid prematurely supposing that we should be hunting around for its "true meaning".

2007.6.27

I should be able to get nice modular nesting of motion commands with zippers if they raise exceptions (or otherwise live in some exception monad) when they can't move any farther, allowing a handler a level up to move to the next higher-level container.

I wonder what differential operator covers the idea of multiple (ordered? nonordered?) 'non-overlapping' holes? The idea is that when looking at the second derivative, the hole taken of the already hole-having datastructure cannot rip out the subtree that has a hole in it.

For a tree whose leaves carry the data, I suppose this is a moot difference. What I'm thinking of is only pertinent for the 'recursive derivative' of a μ-type, which results in the type of lists of the derivative of the body of the μ with respect to its variable.

Is there a way of discovering the power series in D of a linear operator? Assuming it's appropriately 'analytic'? I feel like this thought occurred to me a couple weeks ago. I mean, suppose $F = \sum_i c_i D^i$. Then we can probe F by applying it to monomials:

$$\begin{aligned} Fx^n &= \sum_i c_i D^i x^n \\ &= \sum_i c_i i! \binom{n}{i} x^{n-i} \end{aligned}$$

in particular

$$\begin{aligned} F1 &= c_0 \\ Fx &= c_0 x + c_1 \\ Fx^2 &= c_0 x^2 + 2c_1 x + 2c_2 \\ Fx^3 &= c_0 x^3 + 3c_1 x^2 + 6c_2 x + 6c_3 \end{aligned}$$

so

$$c_1 = Fx - xF1$$
$$c_2 = \frac{Fx^2 - 2xFx + x^2F1}{2}$$
$$c_3 = \frac{Fx^3 - 3xFx^2 + 3x^2Fx - x^3F1}{6}$$

and I would conjecture that

$$c_i = \frac{1}{i!}\sum_j (-1)^j \binom{i}{j} x^j F x^{i-j}$$

Let's see if that works out algebraically:

$$\frac{1}{i!}\sum_j (-1)^j \binom{i}{j} x^j \left(\sum_k c_k k! \binom{i-j}{k} x^{i-j-k}\right)$$
$$= \frac{1}{i!}\sum_j (-1)^j \binom{i}{j} \left(\sum_k c_k k! \binom{i-j}{k} x^{i-k}\right)$$
$$= \frac{1}{i!}\sum_j (-1)^j \frac{i!}{j!(i-j)!} \left(\sum_k c_k k! \frac{(i-j)!}{k!(i-j-k)!} x^{i-k}\right)$$
$$= \sum_j (-1)^j \frac{1}{j!} \left(\sum_k c_k \frac{1}{(i-j-k)!} x^{i-k}\right)$$
$$= \sum_k c_k x^{i-k} \sum_j (-1)^j \frac{1}{j!} \frac{1}{(i-j-k)!}$$
$$= \sum_k c_k \frac{x^{i-k}}{(i-k)!} \sum_j (-1)^j \binom{i-k}{j}$$
$$= \sum_k c_k \frac{x^{i-k}}{(i-k)!} 0^{i-k}$$
$$= c_i$$

Sure does!

* * * * * *

I should also be able to get the converse, that Fx^n is equal to

$$\sum_i \left(\frac{1}{i!}\sum_j (-1)^j \binom{i}{j} x^j F x^{i-j}\right) i! \binom{n}{i} x^{n-i}$$

$$= \sum_i \left(\sum_j (-1)^j \binom{i}{j} x^j F x^{i-j} \right) \binom{n}{i} x^{n-i}$$

$$= \sum_i \sum_j (-1)^j \binom{i}{j} \binom{n}{i} x^{n-(i-j)} F x^{i-j}$$

$m = i - j$, $j = i - m$

$$= \sum_m F x^m \sum_i (-1)^{i-m} \binom{i}{i-m} \binom{n}{i} x^{n-m}$$

$$= \sum_m F x^m \sum_i (-1)^{i-m} \frac{i!}{(i-m)!m!} \frac{n!}{i!(n-i)!} x^{n-m}$$

$$= \sum_m F x^m \frac{n! x^{n-m}}{m!} \sum_i (-1)^{i-m} \frac{1}{(i-m)!(n-i)!}$$

$$= \sum_m F x^m \frac{n! x^{n-m}}{m!} \sum_i (-1)^{i-m} \binom{n-m}{i-m} \frac{1}{(n-m)!}$$

$$= \sum_m F x^m \frac{n! x^{n-m}}{m!(n-m)!} \sum_i (-1)^{i-m} \binom{n-m}{i-m}$$

$$= \sum_m F x^m \frac{n! x^{n-m}}{m!(n-m)!} \sum_i (-1)^i \binom{n-m}{i}$$

$$= \sum_m F x^m \frac{n! x^{n-m}}{m!(n-m)!} 0^{n-m}$$

$$= F x^n \frac{n! x^{n-n}}{n!(n-n)!}$$

$$= F x^n$$

Conclusion: If F is representable as a power series $\sum_i c_i D^i$ in D, then

$$c_i = \sum_j \frac{(-x)^j F(x^{i-j})}{j!(i-j)!}$$

Take for instance $F(P) = P[x/x+1]$.

$$c_i = \sum_j \frac{(-x)^j (x+1)^{i-j}}{j!(i-j)!}$$

$$= \sum_j \frac{(-x)^j}{j!(i-j)!} \sum_k x^k \binom{i-j}{k}$$

$$= \sum_k \sum_j \frac{(-1)^k x^{j+k}}{j!k!(i-j-k)!}$$

$(m = j + k,\ k = m - j)$

$$= \sum_m \frac{x^m}{(i-m)!} \sum_j \frac{(-1)^{m-j}}{j!(m-j)!}$$

$$= \sum_m \frac{x^m}{m!(i-m)!} \sum_j (-1)^{m-j} \binom{m}{j}$$

$$= \sum_m \frac{x^m}{m!(i-m)!} 0^m$$

$$= \frac{1}{i!}$$

So this is of course the multiset-of-holes operator e^D.

2007.6.28

Here's a second attempt at a notion of relativistic CA; the first was sketched in TL from yesterday and also on livejournal. I think it's pretty much nonsense, actually.

Let S be a set of states. For each $n \in \mathbb{N}^+$, let B_n be a map $S^n \to S^n$. E.g., for $n = 3$, think of it as a way of translating states that looks like

$$\square\ \square\ \square \mapsto \begin{matrix}\square\\ \square\\ \square\end{matrix}$$

Also suppose there is an involution $I : S \to S$.

The 'laws of physics' are represented by a function $f : S \times S \to S$. A spacetime tableau $T : \mathbb{Z}^2 \to S$ is valid at $(x, y) : \mathbb{Z}^2$ if $T(x, y) = f(T(x - 1, y), T(x, y - 1))$, and valid (full stop) if it is valid over all $\mathbb{Z}^2$.

The B_n's behavior can be extended to tableaux as follows, for $0 \leq i < n$.

$$(B_n(T))(x, ny + i) = \pi_i B_n(T(nx, y), T(nx + 1, y), \ldots, T(nx + (n - 1), y))$$

Let $\sim$ be the twist map $\lambda(x, y).(y, x)$. We can define an operation $T^{-1} = I \circ T \circ \sim$. We require that $nT = B_n T$ constitutes an action of the multiplicative monoid $\mathbb{N}^+$ on the set of tableaux, and that $(p/q)T =$

$B_p((B_q(T^{-1}))^{-1})$ constitutes an action of the multiplicative group $\mathbb{Q} \setminus \emptyset$ on the set of tableaux, and that it preserves validity in the sense that

$$T \text{ valid} \Leftrightarrow T^{-1} \text{ valid}$$

$$T \text{ valid} \Leftrightarrow B_n(T) \text{ valid}$$

2007.6.29

A third attempt: consider cells as transforming *one-dimensional* data along their bottom and left edges into similarly typed data along their top and right edges. Suppose that we can *locally* boost space by interchanging a row with a column, 'rewiring' things as appropriate, allowing for operators on these bits of one-dimensional data that smoosh (and splice) and stretch (and separate) them as appropriate.

I had a flash of coherent thinking about the completeness of focussing, as pertains to the 'inside-out' induction that I think I tried and failed to apply once upon a time. It should go something like this:

We want to prove a series of theorems, one for each connective, (although I bet strongly that I can make the argument uniform in them but for the important case) like for $\otimes$

If $(\Gamma \vdash J)[d^+r/x^+][r/y]$ then $(\Gamma \vdash J)[d^+p \otimes d^+q/x^+][d^+p \otimes d^+q/y]$

where x^+ is a positive propositional variable, y is a neutral propositional variable, and p, q, r are (neutral) propositional atoms not appearing in $\Gamma \vdash J$. By virtue of the duplicate substitution on the right, the cases for the d^+ rules are trivial! The only interesting case is the neutral init rule, in which case we merely need to cough up an $O(1)$-sized proof of completeness of the particular connective.

We should be able to then build up proofs of the identity property for any compound connective by growing it at the leaves.

2007.6.30

The focussing completeness idea from yesterday is bunk after all; I made a mistake in the 'trivial' case passing from an active judgment to a truth judgment. The only other novel idea I've had lately is replacing already asynchronously decomposed signed atoms $p_\pm$ with 'super-active' judgments that have even higher priority than the usual ones. This is of course rather sketchy, though.

In HLF:

Recall that the synth-checking boundary requires base types, and so sensibly requires exact equality of the synthed and checked type (since they have no worlds in them) but allows slack in the *judgmental* world.

$$\frac{\Gamma \vdash R \Rightarrow (a \cdot S')[p'] \qquad S = S' \qquad p \equiv_{ACU} p'}{\Gamma \vdash R \Leftarrow (a \cdot S)[p]}$$

For the sake of unification I want to add contextual modal variables $u[\sigma]$ to the production for atomic terms R. Thus I should be synthesizing them.

$$\frac{u :: (\Psi \vdash a \cdot S[p]) \in \Delta \qquad \Gamma \vdash \sigma; \sigma^{\mathsf{w}} : \Psi}{\Gamma \vdash u[\sigma] \Rightarrow (a \cdot S[\sigma])[p[\sigma^{\mathsf{w}}]]}$$

Here σ is just a substitution for term variables, and σ^{w} is a substitution for world variables.

$$\frac{\Gamma \vdash M \Leftarrow (A[\sigma; \sigma^{\mathsf{w}}])[\varepsilon] \qquad \Gamma \vdash \sigma; \sigma^{\mathsf{w}} : \Psi}{\Gamma \vdash [M/x]^A, \sigma; \sigma^{\mathsf{w}} : \Psi, x : A}$$

$$\frac{\Gamma \vdash p \Leftarrow \mathsf{w} \qquad \Gamma \vdash \sigma; \sigma^{\mathsf{w}} : \Psi}{\Gamma \vdash \sigma; [p/\alpha], \sigma^{\mathsf{w}} : \Psi, \alpha : \mathsf{w}}$$

$$\frac{}{\Gamma \vdash \mathsf{id}; \mathsf{id} : \cdot}$$

Alternatively I might think of σ as being one big subsitution and being able to pull out σ^{t} its term part and σ^{w} its world part.

$$\frac{u :: (\Psi \vdash a \cdot S[p]) \in \Delta \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash u[\sigma^{\mathsf{t}}] \Rightarrow (a \cdot S[\sigma^{\mathsf{t}}])[p[\sigma^{\mathsf{w}}]]}$$

$$\frac{\Gamma \vdash M \Leftarrow (A[\sigma])[\varepsilon] \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash [M/x]^A, \sigma : \Psi, x : A}$$

$$\frac{\Gamma \vdash p \Leftarrow \mathsf{w} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash [p/\alpha], \sigma : \Psi, \alpha : \mathsf{w}}$$

I would want to show then that if

Lemma 0.4 *All of the following:*

- *If* $\Psi \vdash M \Leftarrow a \cdot S[p]$ *and* $\Gamma \vdash \sigma : \Psi$, *then* $\Gamma \vdash M[\sigma^{\mathsf{t}}] \Leftarrow (a \cdot S[\sigma^{\mathsf{t}}])[p[\sigma^{\mathsf{w}}]]$.
- *If* $\Psi \vdash S : A[p] > (a \cdot S)[q]$ *and* $\Gamma \vdash \sigma : \Psi$, *then* $\Gamma \vdash S[\sigma^{\mathsf{t}}] : A[\sigma][p[\sigma^{\mathsf{w}}]] > (a \cdot S[\sigma^{\mathsf{t}}])[q[\sigma^{\mathsf{w}}]]$.

- *If* $\Psi \vdash R \Rightarrow a \cdot S[p]$ *and* $\Gamma \vdash \sigma : \Psi$, *then* $\Gamma \vdash R[\sigma^{\mathsf{t}}] \Rightarrow (a \cdot S[\sigma^{\mathsf{t}}])[p[\sigma^{\mathsf{w}}]]$.

Proof By induction on the derivation.

Case: $R = x \cdot S_0$ for x such that $\Psi = \Psi, x : A$ and $\sigma = [N/x]^A, \sigma_0$.

$$\frac{x : A \in \Psi \qquad \Psi \vdash S_0 : A[\varepsilon] > a \cdot S[p]}{\Psi \vdash x \cdot S_0 \Rightarrow a \cdot S[p]}$$

By induction hypothesis we get (starting to be sloppy about $\sigma^{\mathsf{t}}, \sigma^{\mathsf{w}}$)

$$\Gamma \vdash S_0[\sigma] : (A[\sigma])[\varepsilon] > a \cdot (S\sigma)[p[\sigma]]$$

But by construction of the substitution we know $N \Leftarrow (A[\sigma_0])[\varepsilon]$ and $A[\sigma_0]$ is the same as $A[\sigma]$ since x can't occur free in A. so we can form a reduction $\Gamma \vdash [N|S_0[\sigma]]^A \Rightarrow a \cdot (S[\sigma])[p[\sigma]]$. And this is what $(x \cdot S_0)[\sigma]$ is equal to anyhow!

■

* * * * * *

Some important things to show:

1. The effect of a simultaneous substitution is the same as unravelling it into sequential substitutions for individual variables.

2. (subsequently) that simultaneous substitutions commute with ordinary substitutions.

3. If $[N/x]^A \in \sigma$, where $\Gamma \vdash \sigma : \Psi$, then $\Gamma \vdash N \Leftarrow \sigma A$. This requires 'weakening' the substitution that applies to the variables actually *in* the type A, up to the whole substitution σ — of course we're only adding things that don't affect A, which is exactly why the lemma holds.

4. The following two expressions are equal:

$$[\lambda x_1 \ldots \lambda x_n . M/u]^{\Pi x_1 : A_1 \ldots o}(u \cdot (M_1; \cdots; M_n))$$

$$[M /\!\!/ u]^{x_n : A_n, \ldots, x_1 : A_1 \vdash o}(u[M_n/x_n, \ldots, M_1/x_1])$$

A meditation on sequences of arguments.
Ordinary application:

$$(\cdots (f\ M_1) \cdots M_n)$$

Spine application:

$$f : \Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n.o \qquad S : \Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n o > o$$

$$f \cdot (M_1; (\cdots M_n; () \cdots))$$

Σ-types:

$$B = (\Sigma x_1{:}A_1 \ldots \Sigma x_n{:}A_n.\top) \qquad f : B \to o \qquad \langle M_1, \langle \cdots M_n, \langle\rangle\rangle \cdots \rangle : B$$

$$f\ \langle M_1, \langle \cdots M_n, \langle\rangle\rangle \cdots \rangle$$

Substitutions:

$$\Psi = (\cdots (x_1 : A_1), \ldots), x_n : A_n \qquad f :: (\Psi \vdash o) \qquad M_n \ldots . M_1.\mathsf{id} : \Psi$$

$$f[M_n \ldots . M_1.\mathsf{id}] : o$$

LF in substitution-form:

$$K ::= \Pi\Psi.\mathrm{type}$$
$$A ::= \Pi\Psi.a[\sigma]$$
$$M ::= \lambda\hat{\Psi}.H[\sigma]$$
$$H ::= c \mid x$$
$$\sigma ::= \mathsf{id} \mid (M/x), \sigma$$
$$\Gamma, \Psi ::= \cdot \mid \Gamma, x : A$$
$$\hat{\Psi} ::= \cdot \mid \Psi, x$$
$$\Sigma ::= \cdot \mid \Sigma, a : K \mid \Sigma, c : A$$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x \Rightarrow A} \qquad \frac{c : A \in \Sigma}{\Gamma \vdash c \Rightarrow A}$$

$$\frac{\Gamma \vdash H \Rightarrow \Pi\Psi.a'[\rho] \qquad \Gamma \vdash \theta : \Psi \qquad a = a' \qquad \sigma = \rho\{\theta\}}{\Gamma \vdash \lambda\hat{\Psi}.H[\theta] \Leftarrow \Pi\Psi.a[\sigma]}$$

$$\frac{\Gamma \vdash \Psi\ \mathrm{ctx} \qquad a : \Pi\Delta.\mathrm{type} \in \Sigma \qquad \Gamma, \Psi \vdash \sigma : \Delta}{\Gamma \vdash \Pi\Psi.a[\sigma] : \mathrm{type}}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Pi\Psi.\text{type} : \text{kind}}$$

$$\frac{}{\Gamma \vdash \mathsf{id} : \cdot} \qquad \frac{\Gamma \vdash M \Leftarrow A\{\sigma\} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash (M/x)^A, \sigma : (\Psi, x : A)}$$

$$\frac{}{\vdash \cdot \text{ ctx}} \qquad \frac{\vdash \Gamma \text{ ctx} \qquad \Gamma \vdash A : \text{type}}{\vdash \Gamma, x : A \text{ ctx}}$$

$$(M_1/x_1, \ldots, M_n/x_n)\sigma = (M_1\{\sigma\}/x_1, \ldots M_n\{\sigma\}/x_n)$$

$$(\lambda\hat{\Psi}.H[\rho])\sigma = \begin{cases} \lambda\hat{\Psi}.H[\rho\{\sigma\}] & \text{if } H \notin \text{dom}(\sigma); \\ \lambda\hat{\Psi}.R\{\rho\{\sigma\}\} & \text{if } H = x \text{ and } \lambda\hat{\Delta}.R/x \in \sigma \end{cases}$$

$$(\Pi\Psi.a[\rho])\sigma = \Pi\Psi.a[\rho\{\sigma\}]$$

Recall that the risk of comparing substitutions for equality is that we might sometimes want to allow noncanonical substitutions to detect patterns and so on. Here, however, it looks okay.

I wonder if I can incorporate base-type polymorphism?

Base Classifiers	v	$::=$	$R \mid \text{base}$
Classifiers	V	$::=$	$\Pi\Psi.v$
Atomic Terms	R	$::=$	$x[\sigma]$
Terms	M	$::=$	$\lambda\hat{\Psi}.R$
Substitutions	σ	$::=$	$\mathsf{id} \mid (M/x)^V, \sigma$
Contexts	Γ	$::=$	$\cdot \mid \Gamma, x : V$

$$\frac{x : \Pi\Psi.v \in \Gamma \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash x[\sigma] \Rightarrow v\{\sigma\}}$$

$$\frac{\Gamma, \Psi \vdash R \Rightarrow v' \qquad v = v'}{\Gamma \vdash \lambda\hat{\Psi}.R \Leftarrow \Pi\Psi.v}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx} \qquad \Gamma, \Psi \vdash R \Rightarrow \text{base}}{\Gamma \vdash \Pi\Psi.R \Leftarrow \text{ok}}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Pi\Psi.\text{base} \Leftarrow \text{ok}}$$

$$\frac{}{\Gamma \vdash \mathsf{id} : \cdot} \qquad \frac{\Gamma \vdash M \Leftarrow V\{\sigma\} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash (M/x)^V, \sigma : (\Psi, x : V)}$$

$$\frac{}{\vdash \cdot \text{ ctx}} \qquad \frac{\vdash \Gamma \text{ ctx} \qquad \Gamma \vdash V \Leftarrow \text{ok}}{\vdash \Gamma, x : V \text{ ctx}}$$

$$(M_1/x_1, \ldots, M_n/x_n)\{\sigma\} = (M_1\{\sigma\}/x_1, \ldots M_n\{\sigma\}/x_n)$$

$$(x[\rho])\{\sigma\} = \begin{cases} R\{\rho\{\sigma\}\} & \text{if } (\lambda\hat{\Psi}.R)/x \in \sigma; \\ x[\rho\{\sigma\}] & \text{otherwise.} \end{cases}$$

$$(\Pi\Psi.R)\{\sigma\} = \Pi\Psi.(R\{\sigma\})$$

$$(\lambda\hat{\Psi}.R)\{\sigma\} = \lambda\hat{\Psi}.(R\{\sigma\})$$

$$\text{base}\{\sigma\} = \text{base}$$

2007.7.1

In DeBruijn form: (still highly incomplete)

Base Classifiers	v	::=	$R \mid \text{base}$
Classifiers	V	::=	$\Pi\Psi.v$
Atomic Terms	R	::=	$n[\sigma]$
Terms	M	::=	$\lambda^n R$
Substitutions	σ	::=	$\mathsf{id} \mid M.\sigma$
Contexts	Γ, Ψ	::=	$\cdot \mid \Gamma, V$

(deBruijn indices are 0-based)

$$\frac{\uparrow^{n+1}(\Gamma(n)) = \Pi\Psi.v \qquad \Gamma \vdash \rho : \Psi}{\Gamma \vdash n[\rho] \Rightarrow v\{\rho\}}$$

$$\frac{\Gamma, \Psi \vdash R \Rightarrow v' \qquad v = v'}{\Gamma \vdash \lambda^{|\Psi|} R \Leftarrow \Pi\Psi.v}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx} \qquad \Gamma, \Psi \vdash R \Rightarrow \text{base}}{\Gamma \vdash \Pi\Psi.R \Leftarrow \text{ok}}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Pi\Psi.\text{base} \Leftarrow \text{ok}}$$

$$\frac{}{\Gamma \vdash \mathsf{id} : \cdot} \qquad \frac{\Gamma \vdash M \Leftarrow V\{\sigma\} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash (M.\sigma) : (\Psi, V)}$$

$$\frac{}{\Gamma \vdash \cdot \text{ ctx}} \qquad \frac{\Gamma \vdash \Psi \text{ ctx} \qquad \Gamma, \Psi \vdash V \Leftarrow \text{ok}}{\Gamma \vdash \Psi, V \text{ ctx}}$$

$$\{\sigma\} = \{\sigma\}_0$$

$$(\Gamma, V)\{\sigma\}_n = (\Gamma\{\sigma\}_n), V\{\sigma\}_{n+|\Gamma|}$$

$$(M.\rho)\{\sigma\}_n = (M\{\sigma\}_n.\rho\{\sigma\}_n)$$

$$(n[\rho])\{\sigma\}_m = \begin{cases} (n-|\sigma|)[\rho\{\sigma\}_m] & \text{if } n-m \geq |\sigma| \\ (\uparrow_k^m R)\{\rho\{\sigma\}_m\} & \text{if } \sigma(n-m) = \lambda^k R; \\ n[\rho\{\sigma\}_m] & \text{if } n-m < 0. \end{cases}$$

$$(\Pi\Psi.R)\{\sigma\}_n = \Pi(\Psi\{\sigma\}_n).(R\{\sigma\}_{n+|\Psi|})$$

$$(\lambda^m R)\{\sigma\}_n = \lambda^m(R\{\sigma\}_{n+m})$$

$$\text{base}\{\sigma\}_n = \text{base}$$

Lemma 0.5 *If* $\sigma(n) = M$, *and* $\Psi(n) = V$, *and* $\Gamma \vdash \sigma : \Psi$, *then* $\Gamma \vdash M \Leftarrow (\uparrow^n V)\{\sigma\}$.

Proof By induction.

Case: $n = 0$. Immediate.

Case: $n = m + 1$. Then σ has the form $(M'.\sigma_0)$, and $\sigma_0(m) = M$. Ψ has the form Ψ_0, V', and $\Psi_0(m) = V$.

$$\frac{\Gamma \vdash M' \Leftarrow V'\{\sigma\}_0 \qquad \Gamma \vdash \sigma_0 : \Psi_0}{\Gamma \vdash (M'.\sigma_0) : (\Psi_0, V')}$$

By induction hypothesis, $\Gamma \vdash M \Leftarrow (\uparrow^m V)\{\sigma_0\}$. To show: $\Gamma \vdash M \Leftarrow (\uparrow^{m+1} V)\{M'.\sigma_0\}$, but this follows.

■

Lemma 0.6 *If*

$$\Gamma, \Delta, \Gamma', \Psi \vdash v \Leftarrow \text{ok} \qquad |\Gamma'| = m$$

$$\Gamma, \Delta, \Gamma' \vdash \rho : \Psi \qquad \Gamma \vdash \sigma : \Delta$$

then $v\{\rho\}\{\sigma\}_m = v\{\sigma\}_{m+|\rho|}\{\rho\{\sigma\}_m\}$

Lemma 0.7 (Substitution) *Suppose* $\Gamma, \Delta, \Gamma' \vdash J$ *and* $\Gamma \vdash \sigma : \Delta$. *Then* $\Gamma, \Gamma'\{\sigma\}_0 \vdash J\{\sigma\}_{|\Gamma'|}$.

$$\frac{\uparrow^{n+1}(\Gamma(n)) = \Pi\Psi.v \qquad \Gamma \vdash \rho : \Psi}{\Gamma \vdash n[\rho] \Rightarrow v\{\rho\}}$$

$$(n[\rho])\{\sigma\}_m = \begin{cases} (n-|\sigma|)[\rho\{\sigma\}_m] & \text{if } n-m \geq |\sigma| \\ (\uparrow_k^m R)\{\rho\{\sigma\}_m\}_0 & \text{if } \sigma(n-m) = \lambda^k R; \\ n[\rho\{\sigma\}_m] & \text{if } n-m < 0. \end{cases}$$

consider the variable case of this theorem:

$$\frac{\uparrow^{n+1}((\Gamma,\Delta,\Gamma')(n)) = \Pi\Psi.v \qquad \Gamma,\Delta,\Gamma' \vdash \rho : \Psi}{\Gamma,\Delta,\Gamma' \vdash n[\rho] \Rightarrow v\{\rho\}}$$

I have already by the induction hypothesis that (with $m = |\Gamma'|$)

$$\Gamma,\Gamma'\{\sigma\} \vdash \rho\{\sigma\}_m : \Psi\{\sigma\}_m$$

and I want to show

$$\Gamma,\Gamma'\{\sigma\} \vdash n[\rho]\{\sigma\}_m \Rightarrow v\{\rho\}\{\sigma\}_m$$

in the first case, $n \geq m + |\sigma|$, so $(\Gamma,\Delta,\Gamma')(n) = \Gamma(n - m - |\sigma|) = (\Gamma,\Gamma'\{\sigma\})(n - |\sigma|)$. Rule application yields

$$\frac{\uparrow^{n-|\sigma|+1}((\Gamma,\Gamma'\{\sigma\})(n-\sigma)) = \Pi\Psi'.v' \qquad \Gamma,\Gamma'\{\sigma\} \vdash \rho\{\sigma\}_m : \Psi'}{(\Gamma,\Gamma'\{\sigma\})(n-|\sigma|)[\rho\{\sigma\}_m] \Rightarrow v'\{\rho\{\sigma\}_m\}}$$

We know that $\uparrow^{|\sigma|}(\Pi\Psi'.v') = \Pi\Psi.v$???

$$\frac{(\Gamma,\Gamma'\{\sigma\}_0)(n-|\sigma|) = \Pi\Delta.v \qquad \Gamma,\Gamma'\{\sigma\}_0 \vdash \rho\{\sigma\}_m : \Delta\{\sigma\}_m}{\Gamma,\Delta,\Gamma' \vdash n[\rho] \Rightarrow v\{\rho\}}$$

in the second case, $(\Gamma,\Psi,\Gamma')(n) = \Psi(n-m) = \Pi\Delta.v$. Here $k = |\Delta|$. I also know that $\sigma(n-m) = \lambda^k R$. By the previous lemma, $\Gamma \vdash \lambda^k R \Leftarrow (\uparrow^{n-m}\Pi\Delta.v)\{\sigma\}$. In other words $\Gamma \vdash \lambda^k R \Leftarrow (\Pi\Delta.\uparrow^{n-m}{}_k v\{\sigma\}_k)$. By inversion

$$\Gamma,\Delta \vdash R \Rightarrow \uparrow^{n-m}{}_k v\{\sigma\}_k$$

we seem to be able to shift to get

$$\Gamma,\Gamma'\{\sigma\},\Delta \vdash \uparrow^m_k R \Rightarrow \uparrow^n{}_k v\{\sigma\}_k$$

now substitute:

$$\frac{\begin{array}{c}\Gamma,\Gamma'\{\sigma\},\Delta \vdash \uparrow^m_k R \Rightarrow \uparrow^n{}_k v\{\sigma\}_k \\ (\Gamma,\Gamma'\{\sigma\}_0)(n-|\sigma|) = \Pi\Delta.v \qquad \Gamma,\Gamma'\{\sigma\}_0 \vdash \rho\{\sigma\}_m : \Delta\{\sigma\}_m\end{array}}{\Gamma,\Gamma'\{\sigma\}_0 \vdash (\uparrow^m_k R)\{\rho\{\sigma\}_m\}_0 \Rightarrow (\uparrow^{n+1}_k v)\{\rho\}_0\{\sigma\}_{|\Gamma'|}}$$

in the third case, $n < m$, so $(\Gamma,\Psi,\Gamma')(n) = \Gamma'(n)$. Rule application yields

$$\frac{(\Gamma,\Gamma'\{\sigma\}_0)(n) = (\Pi\Delta.v)\{\sigma\}_{m-n-1} \qquad \Gamma,\Gamma'\{\sigma\}_0 \vdash \rho\{\sigma\}_m : \Delta\{\sigma\}_m}{(\Gamma,\Gamma'\{\sigma\}_0) \vdash n[\rho\{\sigma\}_m] \Rightarrow (\uparrow^{n+1}_{|\Delta|} v\{\sigma\}_{m-n-1+|\Delta|})\{\rho\{\sigma\}_m\}_0}$$

$$
\begin{array}{rlll}
\text{Base Classifiers} & v & ::= & R \mid \mathsf{type} \\
\text{Classifiers} & V & ::= & \Pi\Psi.v \\
\text{Atomic Terms} & R & ::= & x[\sigma] \\
\text{Terms} & M & ::= & \lambda\hat{\Psi}.R \\
\text{Substitutions} & \sigma & ::= & \mathsf{id} \mid M.\sigma \\
\text{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x : V
\end{array}
$$

$$
\frac{x : \Pi\Psi.v \in \Gamma \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash x[\sigma] \Rightarrow v\{\sigma/\Psi\}}
$$

$$
\frac{\Gamma, \Psi \vdash R \Rightarrow v' \qquad v = v'}{\Gamma \vdash \lambda\hat{\Psi}.R \Leftarrow \Pi\Psi.v}
$$

$$
\frac{\Gamma \vdash \Psi\ \mathrm{ctx} \qquad \Gamma, \Psi \vdash R \Rightarrow \mathsf{type}}{\Gamma \vdash \Pi\Psi.R \Leftarrow \mathrm{ok}}
$$

$$
\frac{\Gamma \vdash \Psi\ \mathrm{ctx}}{\Gamma \vdash \Pi\Psi.\mathsf{type} \Leftarrow \mathrm{ok}}
$$

$$
\frac{}{\Gamma \vdash \mathsf{id} : \cdot} \qquad \frac{\Gamma \vdash M \Leftarrow V\{\sigma/\Psi\} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash M.\sigma : (\Psi, x : V)}
$$

$$
\frac{}{\Gamma \vdash \cdot\ \mathrm{ctx}} \qquad \frac{\Gamma \vdash \Psi\ \mathrm{ctx} \qquad \Gamma, \Psi \vdash V \Leftarrow \mathrm{ok}}{\Gamma \vdash \Psi, x : V\ \mathrm{ctx}}
$$

$$
(M_1 \ldots M_n.\mathsf{id})\{\sigma/\Psi\} = (M_1\{\sigma/\Psi\} \ldots M_n\{\sigma/\Psi\}.\mathsf{id})
$$

$$
(V_1, \ldots, V_n)\{\sigma/\Psi\} = (V_1\{\sigma/\Psi\}, \ldots, V_n\{\sigma/\Psi\})
$$

$$
(x[\rho])\{\sigma/\Psi\} = \begin{cases} R\{\rho\{\sigma/\Psi\}/\Delta\} & \text{if } \Psi(n) = x : \Pi\Delta.V \text{ and } \sigma(n) = \lambda\hat{\Delta}.R; \\ x[\rho\{\sigma/\Psi\}] & \text{otherwise.} \end{cases}
$$

$$
(\Pi\Delta.v)\{\sigma/\Psi\} = \Pi\Delta.(v\{\sigma/\Psi\})
$$

$$
(\lambda\hat{\Delta}.R)\{\sigma/\Psi\} = \lambda\hat{\Delta}.(R\{\sigma/\Psi\})
$$

$$
\mathsf{type}\{\sigma/\Psi\} = \mathsf{type}
$$

Don't know whether to treat worlds as being really modal, or else just talking about restriction to certain contexts. The latter sounds easier...

2007.7.2

By the equivalence principle, an accelerated reference frame behaves locally the same as a frame at rest under a uniform gravitational field. Is

there any value to the intuition that this means massive objects act as if they are expanding and therefore accelerating towards us? Under appropriate rescaling, can gravitational attraction be construed not as warping of spacetime, but progressive *consumption* of it?

2007.7.3

Using the same tricks as are used in η-expansion for polymorphism, I think I can get away with n-ary homogeneous tuples that actually allow projection. The analogue of the trick is that if I have a variable x whose type is a vector of type a (which *is* required to be a base type) of length $s^n(i)$, where $i : nat$ is a variable, then x's η-expanded canonical form is

$$\mathsf{hd}\, x :: \mathsf{hd}\, \mathsf{tl}\, x :: \cdots :: \mathsf{hd}\, \mathsf{tl}^{n-1}\, x :: \eta_i(\mathsf{tl}^n\, x)$$

where the understanding is that hereditary substitution does things like

$$[z/i]\eta_i(R) = []$$

$$[s\ n/i]\eta_i(R) = \mathsf{hd}\, R :: \eta_n(\mathsf{tl}\, R)$$

The restriction the list carrier type to base types ensures that I don't have to bother about actually η-expanding *elements* of the list as well, which would frightfully intertwine the term and type syntaxes, as is actually done in the case of Nanevski-Morrisett-Birkedal polymorphism.

2007.7.4

So if I try to put a series of modal boxes back into 'substitution-form' LF, I don't seem to get the restriction of local contexts present in Frank's description of how he tried to close CMTT up to ω. The only changes required seem to be

$$\text{Contexts}\quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x :_n V$$

$$\frac{x :_n \Pi\Psi.v \in \Gamma \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash x[\sigma] \Rightarrow v\{\sigma/\Psi\}}$$

$$\frac{\Gamma\big|_{\geq n} \vdash M \Leftarrow V\{\sigma/\Psi\} \qquad \Gamma \vdash \sigma : \Psi}{\Gamma \vdash M.\sigma : (\Psi, x :_n V)}$$

$$\frac{\Gamma \vdash \Psi\ \text{ctx} \qquad \Gamma, \Psi \vdash V \Leftarrow \text{ok}}{\Gamma \vdash \Psi, x :_n V\ \text{ctx}}$$

No, hm, now that I think about it, the substitution formation rule should restrict the remaining substitution also (and the context formation

rule should restrict some part of the context when checking V, though I'm not sure how much) to maintain the invariant that we only type-check against well-formed types. This might impose the constraint that any particular context pragmatically needs to be ordered by modality strength in order to make substitutions possible.

* * * * * *

I realize why the context formation rule has to in fact be the very restrictive

$$\frac{\Gamma\big|_n \vdash \Psi \text{ ctx} \qquad (\Gamma, \Psi)\big|_n \vdash V \Leftarrow \text{ok}}{\Gamma \vdash \Psi, x :_n V \text{ ctx}}$$

It's because the following should obviously be true: if Γ is a valid context, then so too is $\Gamma\big|_n$. Therefore n-strong hypotheses should not be able to having types depending on weaker ones, for the latter might disappear during a restriction.

* * * * * *

Also, hey, I seem to get a form of negation in these base-type polymorphism frameworks! How weird.

$$\neg A = A \to (\Pi x{:}\text{type}.x)$$

I only get to eliminate it at base types, but I can lift it to function types. Let's try to do this in substitution form:

$$\neg A = \Pi(x : A, y : \text{type}).y[]$$

So for instance I can still prove $x : A \vdash M : \neg\neg A$ by

$$M = \lambda(x_0, t_0).x_0[\eta_A(x), t_0[]]$$

and if $x : A \vdash M : B$ then $y : \neg B \vdash N : \neg A$ by

$$N = \lambda(x, t).y[M, t[]]$$

composing these and setting $B = \neg\neg A$ we get

$$y : \neg\neg\neg A \vdash \lambda(x, t).y[\lambda(x_0, t_0).x_0[\eta_A(x), t_0[]], t[]] : \neg A$$

Whoops, I guess I never actually needed to eliminate that falsehood at anything other than base types. What about disjunction?

$$A \vee B = \Pi x{:}\text{type}.(A \to x) \to (B \to x) \to x$$

Let a type $C \to D$ be given. Can we do this?

$$(A \vee B) \to (A \to C \to D) \to (B \to C \to D) \to C \to D$$

It seems so.

$$\lambda d b_1 b_2 c.\ d\ D\ (\lambda x.b_1\ x\ c)\ (\lambda x.b_2\ x\ c)$$

So this kind of thing will obey β but basically no η laws. And of course it seriously infects every base type with eliminations!

2007.7.5

$$(m[\rho])\{\sigma\}_n = \begin{cases} m[\rho\{\sigma\}_n] & \text{if } m-n<0 \\ R\uparrow_k^n\{\rho\{\sigma\}_n\} & \text{if } \sigma(m-n)=\lambda^k R \\ (m-|\sigma|)[\rho\{\sigma\}_n] & \text{if } m-n\geq|\sigma| \end{cases}$$

Lemma 0.8 $X\uparrow\{M.\sigma\} = X\{\sigma\}$

Lemma 0.9 $X\uparrow_m\{\sigma\}_{1+n} = X\{\sigma\}_n\uparrow_m$

Lemma 0.10 $X\uparrow_{k+m}^m\{\sigma\}_{k+m+n} = X\{\sigma\}_{k+n}\uparrow_{k+m}^m$

Lemma 0.11 *If* $\Gamma,\Delta\vdash J$ *then* $\Gamma,\Psi,\Delta\uparrow^{|\Psi|}\vdash J\uparrow_{|\Delta|}^{|\Psi|}$.

Lemma 0.12 *If* $\Gamma(m)=V$ *then* $(\Gamma\{\sigma\})(m) = V\{\sigma\}^{|\Gamma|-m-1}$.

Lemma 0.13 *If* $\Psi(m)=V$ *and* $\Gamma\vdash\sigma\Leftarrow\Psi$ *then* $\Gamma\vdash\sigma(m)\Leftarrow V\uparrow^{m+1}\{\sigma\}$.

$$\frac{\Gamma(m)=\Pi\Delta.v \qquad \Gamma\vdash\rho\Leftarrow\Delta\uparrow^{m+1}}{\Gamma\vdash m[\rho]\Rightarrow v\uparrow_{|\Delta|}^{m+1}\{\rho\}}$$

Theorem 0.14 *If* $\Gamma,\Psi,\Gamma'\vdash J$ *and* $\Gamma\vdash\sigma\Leftarrow\Psi$, *then* $\Gamma,\Gamma'\{\sigma\}\vdash J\{\sigma\}_{|\Gamma'|}$.

Proof By induction.

Case: Abbreviate $n=|\Gamma'|$, and $s=|\Psi|=|\sigma|$ and $k=|\Delta|=|\rho|$. The derivation of J is

$$\frac{(\Gamma,\Psi,\Gamma')(m)=\Pi\Delta.v \qquad \Gamma,\Psi,\Gamma'\vdash\rho\Leftarrow\Delta\uparrow^{m+1}}{\Gamma,\Psi,\Gamma'\vdash m[\rho]\Rightarrow v\uparrow_k^{m+1}\{\rho\}}$$

By i.h., we have $\Gamma,\Gamma'\{\sigma\}\vdash\rho\{\sigma\}_n\Leftarrow\Delta\uparrow^{m+1}\{\sigma\}_n$. We must show

$$\Gamma,\Gamma'\{\sigma\}\vdash(m[\rho])\{\sigma\}_n\Rightarrow v\uparrow_k^{m+1}\{\rho\}\{\sigma\}_n$$

Subcase: $m<n$.

$(\Gamma, \Psi, \Gamma')(m) = \Pi\Delta.v$ Assumption
$\Gamma'(m) = \Pi\Delta.v$
$\Gamma'\{\sigma\}(m) = (\Pi\Delta.v)\{\sigma\}^{n-m-1}$ Lemma 0.12
$(\Gamma, \Gamma'\{\sigma\})(m) = (\Pi\Delta.v)\{\sigma\}^{n-m-1}$

$$\dfrac{(\Gamma, \Gamma'\{\sigma\})(m) = (\Pi\Delta.v)\{\sigma\}_{n-m-1} \quad \dfrac{\Gamma, \Gamma'\{\sigma\} \vdash \rho\{\sigma\}_n \Leftarrow \Delta\uparrow^{m+1}\{\sigma\}_n}{\Gamma, \Gamma'\{\sigma\} \vdash \rho\{\sigma\}_n \Leftarrow \Delta\{\sigma\}_{n-m-1}\uparrow^{m+1}} =}{\dfrac{\Gamma, \Gamma'\{\sigma\} \vdash m[\rho\{\sigma\}_n] \Rightarrow v\{\sigma\}_{k+n-m-1}\uparrow_k^{m+1}\{\rho\{\sigma_n\}\}}{\dfrac{\Gamma, \Gamma'\{\sigma\} \vdash m[\rho\{\sigma\}_n] \Rightarrow v\uparrow_k^{m+1}\{\sigma\}_{k+n}\{\rho\{\sigma\}_n\}}{\Gamma, \Gamma'\{\sigma\} \vdash m[\rho\{\sigma\}_n] \Rightarrow v\uparrow_k^{m+1}\{\rho\}\{\sigma\}_n} =} =}$$

Subcase: $n \leq m < n + s$.

Subcase: $m \geq n + s$.

■

2007.7.6

In the code to check substitution-form LF, I do something like

```
fun ok_base G r = r = (~1,[]) orelse synth G r = (~1,[])
```

Would it be terminating to check in a while loop if r synths to something that synths to something that $\cdots$ synths to type? I'm pretty sure it does, for the head of a classifier of any variable has to exist in a smaller context.

2007.7.7

It's questionable how useful the tower of hyper^{n}kinds actually is, given the restriction to base everythings. Still kind of interesting. Multimodal stuff works out great, and proof irrelevance seems to also; not certain the full extent of how they can be combined. Taking their cartesian product seems okay, but definitely not to have modal restriction turn disqualified modal variables into irrelevant ones.

2007.7.8

Apparently the conditions put on Hilbert spaces make them unique such that they are isomorphic to their dual spaces. Awfully nice, that!

2007.7.9

The quantum harmonic oscillator is starting to make some sense. I still haven't internalized what the ladder operators look like in position space, but they're super-spookily suggestive in count space of how differentiation and 'times x' work on generating functions.

2007.7.10

In GR, it's still not clear to me how you coordinatize the curvature vector.

DeBruijnifying substitution-style LF does not seem at all easier than otherwise, which is annoying.

2007.7.11

If a state is $z = kq + ip$ then hamilton's equations are

$$kH_z = i\dot{z}^*$$

2007.7.12

Fully-contextual modal stuff looks cute in simple types:

$$\begin{array}{rrcl} \text{Props} & A & ::= & \Gamma \to p \\ \text{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, A^n \end{array}$$

$$\frac{\Gamma, \Psi \vdash p}{\Gamma \vdash \Psi \to p} \qquad \frac{\Gamma \vdash \Psi}{\Gamma, (\Psi \to p)^n \vdash p}$$

$$\frac{\Gamma\big|_n \vdash A \qquad \Gamma \vdash \Psi}{\Gamma \vdash \Psi, A^n} \qquad \frac{}{\Gamma \vdash \cdot}$$

Alternatively I could get it down to 3 rules like:

$$\frac{\Gamma \vdash \Psi}{\Gamma, (\Psi \to p)^n \vdash p}$$

$$\frac{\Gamma\big|_n, \Omega \vdash p \qquad \Gamma \vdash \Psi}{\Gamma \vdash \Psi, (\Omega \to p)^n} \qquad \frac{}{\Gamma \vdash \cdot}$$

Or by cheating, 2 rules:

$$\frac{\Gamma \vdash \Psi}{\Gamma, (\Psi \to p)^n \vdash p}$$

$$\frac{\Gamma\big|_{n_i}, \Omega_i \vdash p_i}{\Gamma \vdash (\Omega_i \to p_i)^{n_i}}$$

Or even 1 rule!

$$\frac{\forall (\Omega \to q)^m \in \Psi . (\Gamma\big|_m, \Omega \vdash q)}{\Gamma, (\Psi \to p)^n \vdash p}$$

Modal without contextual:

$$\begin{array}{rrcl} \text{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, A^n \\ \text{Props} & A & ::= & p \mid A \to_n B \end{array}$$

$$\frac{\Gamma, A^n \vdash B}{\Gamma \vdash A \to_n B} \qquad \frac{\Gamma\big|_n \vdash A \qquad \Gamma, B^0 \vdash C}{\Gamma, (A \to_n B)^m \vdash C}$$

$$\frac{}{\Gamma, p^0 \vdash p}$$

2007.7.13

Finally figured out how to *derive* relativistic momentum and mass from Lorentz invariance of the Lagrangian.

Say we've got $L(q, \dot{q})$. First of all assume it's independent of q, and only depends on the magnitude of $\dot{q}$. So we've got some function $L(v)$. Consider the following two (Lorentz-equivalent) scenarios. Scenario (I): A particle at the origin sits around at rest for one time-unit, reaching event $(1, 0)$. Scenario (II) (Lorentz-transformed from (I) by velocity v) a particle moves inertially from the origin to the event $\gamma(1, v)$. We are abbreviating $\gamma = \frac{1}{\sqrt{1-v^2}}$ as customary. Computing the Lagrangian integral for (I) and (II) and setting them equal we get

$$\int_{t=0}^{t=1} L(0) = \int_{t=0}^{t=\gamma} L(v)$$

but this means

$$L(0) = \gamma L(v)$$

so $L(v) = L(0)/\gamma$. We know what the Lagrangian must be for all velocities, assuming we know what it is for $v = 0$.

Now we *define* momentum to be the $\dot{q}$-derivative of the langrangian, and define mass to be momentum divided by velocity. We easily get

$$p = L_{\dot{q}} = \frac{d}{dv} L(0)\sqrt{1 - v^2} = \frac{L(0)v}{\sqrt{1 - v^2}} = L(0)\gamma v$$

and

$$m = p/v = L(0)\gamma$$

Clearly $L(0)$ is just the rest-mass of the particle.

2007.7.14

I think I finally believe in the equivalence of provability-expressivity of intuitionistic and classical multimodal logic. The key to the proof being sensible is precisely a focusing discipline for both systems.

Intuitionistic system:

$$\begin{array}{rrcl} \text{Contexts} & \Gamma, \Psi, \Omega & ::= & \cdot \mid \Gamma, A^n \\ \text{Props} & A & ::= & \Psi \to p \end{array}$$

$$\frac{\Gamma \vdash \Psi}{\Gamma, (\Psi \to p)^n \vdash p} \qquad \frac{\forall (\Omega \to p)^n \in \Psi.\ (\Gamma\big|_n, \Omega \vdash p)}{\Gamma \vdash \Psi}$$

Classical system:

$$\begin{array}{rrcl} \text{Contexts} & \Delta, \Xi & ::= & \cdot \mid \Delta, C^n \\ \text{Props} & C & ::= & p \mid p^\perp \mid \bigwedge \Xi \mid \bigvee \Xi \end{array}$$

$$\frac{}{\Delta, p, p^\perp \Downarrow} \qquad \frac{\Delta, C \Downarrow}{\Delta, C^n \Downarrow}$$

$$\frac{\Delta, \Xi \Downarrow}{\Delta, \bigwedge \Xi \Downarrow} \qquad \frac{\forall C^n \in \Xi.\ (\Delta\big|_n, C \Downarrow)}{\Delta, \bigvee \Xi \Downarrow}$$

Double-negation translation:

$$\boxed{\Delta \Downarrow \quad \Leftrightarrow \quad \Delta^* \vdash \star}$$

$$\neg \Psi = \Psi \to \star$$

$$\sim (C^n, \ldots) = ((\neg C)^n, \ldots)$$

$$(\bigwedge \Xi)^* = \neg\neg(\Xi^*)$$

$$(\bigvee \Xi)^* = \neg \sim (\Xi^*)$$

$$p^* = p$$

$$(p^\perp)^* = \neg p$$

The rule $\dfrac{\Delta^*, C \vdash \star}{\Delta^*, C^n \vdash \star}$ is admissible. Proof analogs of other rules go like

$$\frac{\dfrac{}{p \vdash p}}{\Delta^*, p, p \to \star \vdash \star} \qquad \frac{\dfrac{\Delta^*, \Xi^* \vdash \star}{\Delta \vdash \neg(\Xi^*)}}{\Delta, \neg\neg(\Xi^*) \vdash \star}$$

$$\frac{\dfrac{\forall C^n \in \Xi.\ (\Delta^*\big|_n, C^* \vdash \star)}{\Delta^* \vdash \sim(\Xi^*)}}{\Delta^*, \neg \sim (\Xi^*) \vdash \star}$$

Box-translation:

$$x ::= t \mid f$$

$$\boxed{\begin{array}{rcl} \Gamma^t, p^\perp \Downarrow & \Leftrightarrow & \Gamma \vdash p \\ \Gamma^t, \bigvee \Psi^f \Downarrow & \Leftrightarrow & \Gamma \vdash \Psi \end{array}}$$

$$(A_1^{m_1}, \ldots, A_n^{m_n})^x = ((A_1^x)^{m_1+1}, \ldots, (A_n^x)^{m_n+1})$$

$$(\Psi \to p)^t = \bigvee(\bigvee \Psi^f, p)$$

$$(\Psi \to p)^f = \bigwedge(\Psi^t, p^\perp)$$

Classical proof analogs of intuitionistic rules go like

$$\dfrac{\dfrac{\Gamma^t, \bigvee \Psi^f \Downarrow \qquad \overline{p, p^\perp \Downarrow}}{\Gamma^t, \bigvee(\bigvee \Psi^f, p), p^\perp \Downarrow}}{\Gamma^t, \bigvee(\bigvee \Psi^f, p)^n, p^\perp \Downarrow}$$

$$\dfrac{\dfrac{\forall(\bigwedge(\Omega^t, p^\perp))^{n+1} \in \Psi^f.\ (\Gamma^t\big|_{n+1}, \Omega^t, p^\perp \Downarrow)}{\forall(\bigwedge(\Omega^t, p^\perp))^{n+1} \in \Psi^f.\ (\Gamma^t\big|_{n+1}, \bigwedge(\Omega^t, p^\perp) \Downarrow)}}{\Gamma^t, \bigvee \Psi^f \Downarrow}$$

The important thing about this second proof is that $\bigvee \Psi^f$ *cannot* survive to the top right, because of modal exclusion.

Hmm... Actually, since to complete the proof I would have to take advantage of the invertibility of $\bigvee$ anyway, I basically want the induction hypothesis to actually be

$$\boxed{\begin{array}{rcl} \Gamma^t, p^\perp \Downarrow & \Leftrightarrow & \Gamma \vdash p \\ \forall A^n \in \Psi.(\Gamma^t\big|_{n+1}, A^f \Downarrow) & \Leftrightarrow & \Gamma \vdash \Psi \end{array}}$$

which would allow me to restore the more symmetric definition

$$(\Psi \to p)^t = \bigvee(\Psi^f, p)$$

Alternative classical system:

$$\begin{array}{rrcl} \text{Contexts} & \Delta, \Xi & ::= & \cdot \mid \Delta, J \\ \text{Judgments} & J & ::= & C\ \mathsf{t}^n \mid C\ \mathsf{f}^n \\ \text{Props} & C & ::= & p \mid \bigwedge \Xi \end{array}$$

$$\frac{}{\Delta, p\ \mathsf{t}, p\ \mathsf{f} \Downarrow} \qquad \frac{\Delta, C\ \mathsf{t} \Downarrow}{\Delta, C\ \mathsf{t}^n \Downarrow} \qquad \frac{\Delta\big|_n, C\ \mathsf{f} \Downarrow}{\Delta, C\ \mathsf{f}^n \Downarrow}$$

$$\frac{\Delta, \Xi \Downarrow}{\Delta, \bigwedge \Xi\ \mathsf{t} \Downarrow} \qquad \frac{\forall J \in \Xi.\ (\Delta, \tilde{J} \Downarrow)}{\Delta, \bigwedge \Xi\ \mathsf{f} \Downarrow}$$

Hm, this doesn't actually seem any simpler.

Classical Logical Framework?

$$\begin{array}{rrcl}
\text{Contexts} & \Delta, \Xi & ::= & \cdot \mid \Delta, x :_n V \\
\text{Substitutions} & \sigma & ::= & \mathsf{id} \mid B.\sigma \\
\text{Classifiers} & V & ::= & v \mid v^\perp \mid \bigwedge \Xi \mid \bigvee \Xi \\
\text{Base Classifiers} & v & ::= & B \mid U_n \\
\text{Branches} & B & ::= & \Xi.E \\
\text{Expressions} & E & ::= & x[\sigma] \mid x[B] \mid y[x]
\end{array}$$

$$\frac{}{\Delta, x : v, y : v^\perp \vdash y[x] \Downarrow}$$

$$\frac{x :_n \bigwedge \Xi \in \Delta \qquad \Delta, \Xi \vdash E \Downarrow}{\Delta \vdash x[\Xi.E] \Downarrow} \qquad \frac{x :_n \bigvee \Xi \in \Delta \qquad \Delta \vdash \sigma : \Xi}{\Delta \vdash x[\sigma] \Downarrow}$$

$$\frac{\Delta\big|_n, u : V \vdash E \Downarrow \qquad \Delta \vdash \sigma : \Upsilon}{\Delta \vdash (u.E).\sigma : (\Upsilon, x :_n V)} \qquad \frac{}{\Delta \vdash \mathsf{id} : \cdot}$$

$$\frac{m \geq n}{\Delta \vdash U_m \Rightarrow \mathrm{class}^n} \qquad \frac{\Delta, u : v^\perp \vdash E \Downarrow \qquad \Delta \vdash v \Rightarrow \mathrm{class}^{n+1}}{\Delta \vdash u.E \Rightarrow \mathrm{class}^n}$$

Maybe disjunctions are not allowed to be dependent? Conjunctions are basically Σs, right? Good lord, what happens if you try to perp a $\bigwedge$, then? Maybe this doesn't work after all.

2007.7.15

The solution to dependent sigmas being hard to refute might just be dependent nand!

$$\text{Classifiers} \quad V \quad ::= \quad v \mid v^\perp \mid \Sigma\Xi \mid N\Xi$$

$$\frac{x : N\Xi \in \Delta \qquad \forall (\Xi', u : V) \leq \Xi.(\Delta, \Xi', u : V^\perp \vdash M_i \Downarrow)}{\Delta \vdash x[\overline{\hat{\Xi}', u.M}] \Downarrow}$$

where $\leq$ means 'is a prefix of'

$$(v^\perp)^\perp = v$$

$$(v)^\perp = v^\perp$$

$$(\Sigma\Psi)^\perp = N\Psi$$

$$(N\Psi)^\perp = \Sigma\Psi$$

Maybe this means splitting the context into true and false assumptions is the right way to go after all.

$$\begin{array}{rlll} \text{Booleans} & b & ::= & \mathsf{t} \mid \mathsf{f} \\ \text{Contexts} & \Delta, \Xi & ::= & \cdot \mid \Delta, x :_{bn} V \\ \text{Substitutions} & \sigma & ::= & \mathsf{id} \mid B, \sigma \\ \text{Classifiers} & V & ::= & v \mid \Sigma\Xi \\ \text{Base Classifiers} & v & ::= & B \mid U_n \\ \text{Branches} & B & ::= & \hat{\Xi}.E \\ \text{Expressions} & E & ::= & x[\sigma] \mid x[B] \mid y[x] \end{array}$$

$$\frac{y :_{\mathsf{f}n} v, x :_{\mathsf{t}n} v \in \Delta}{\Delta \vdash y[x] \Downarrow}$$

$$\frac{x :_{\mathsf{f}n} \Sigma\Xi \in \Delta \qquad \Delta \vdash \sigma : \Xi}{\Delta \vdash x[\sigma] \Downarrow} \qquad \frac{x :_{\mathsf{t}n} \Sigma\Xi \in \Delta \qquad \Delta, \Xi \vdash E \Downarrow}{\Delta \vdash x[\hat{\Xi}.E] \Downarrow}$$

$$\frac{}{\Delta \vdash \mathsf{id} : \cdot} \qquad \frac{\Delta \vdash \sigma : \Xi \qquad \Delta\big|_m, x :_{\bar{b}} V\{\sigma/\Xi\} \vdash E \Downarrow}{\Delta \vdash (x.E), \sigma : (\Xi, u :_{bm} V)}$$

2007.7.16

The remaining thing about classicizing LF that still confuses me is what happens at the classifier level. Since these systems seem rather insensitive to what the base classifiers are, I'm tempted to define classifiers and context validity by something like

$$\begin{array}{rlll} \text{Classifiers} & V & ::= & v \mid \Sigma\Psi \\ \text{Base Classifiers} & v & ::= & (u.E : v) \mid U_n \end{array}$$

$$\frac{(\Gamma, \Psi)\big|_n \vdash V \Leftarrow \text{class}}{\Gamma \vdash \Psi, x :_{bn} V \text{ ctx}}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Sigma\Psi \Leftarrow \text{class}} \qquad \frac{\Gamma \vdash v \Rightarrow^n \text{class}}{\Gamma \vdash v \Leftarrow \text{class}}$$

$$\frac{\Gamma, u :_{\mathsf{f}} v \vdash E \Downarrow \qquad \Gamma \vdash v \Rightarrow^{n+1} \text{class}}{\Gamma \vdash (u.E : v) \Rightarrow^{n} \text{class}}$$

$$\frac{m \geq n}{\Gamma \vdash U_m \Rightarrow^{n} \text{class}}$$

2007.7.17

In fact the apparent insensitivity of much of the metatheory of LF extensions is quite frustrating! I suppose I want to by fiat impose a constraint that type checking should maintain the invariant that all contexts and input types should be valid, but it's not clear where this constraint comes from. If one was more liberal about allowing the presence of ill-formed types, what damage would it do to adequacy theorems?

I would conjecture that the answer is 'none' but clearly we want kind-checking in day-to-day Twelf hacking. If I write down a clause of a meta-proof, then it being ill-typed is fairly disastrous. Already I want to use the refinement ability that dependent types give me at the kind level to provide a sanity check on my definitions.

2007.7.18

$$\frac{\Gamma, \Psi \vdash R \Rightarrow v}{\Gamma \vdash \lambda\hat{\Psi}.R \Leftarrow \Pi\Psi.v}$$

$$\frac{x :_j \Pi\Psi.v \in \Gamma \qquad \Gamma \vdash \sigma \Leftarrow \Psi \qquad hyp(j)}{\Gamma \vdash x[\sigma] \Rightarrow v\{\sigma/\Psi\}}$$

$$\frac{\Gamma@j \vdash M \Leftarrow V\{\sigma/\Psi\} \qquad \Gamma \vdash \sigma \Leftarrow \Psi}{\Gamma \vdash M.\sigma \Leftarrow \Psi, x :_j V}$$

$$\frac{}{\Gamma \vdash \text{type} \Rightarrow \text{class}} \qquad \frac{\Gamma \vdash R \Rightarrow \text{type}}{\Gamma \vdash R \Rightarrow \text{class}}$$

$$\frac{(\Gamma, \Psi) * j \vdash V \Leftarrow \text{class} \qquad \Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Psi, x :_j V \text{ ctx}} \qquad \frac{\Gamma, \Psi \vdash v \Rightarrow \text{class} \qquad \Gamma \vdash \Psi \text{ ctx}}{\Gamma \vdash \Pi\Psi.v \Leftarrow \text{class}}$$

Claim: we should only type-check in valid contexts, against valid types.
Claim: $\vdash \Gamma$ ctx and $\Gamma \vdash \Psi$ ctx iff $\vdash \Gamma, \Psi$ ctx

Postulate Let $\star \in \{@, *\}$.
$\leq$ is reflexive.
$i * j \leq (i \star k) * (j \star k)$

If $i \leq j$ and $hyp(i)$ then $hyp(j)$.
Both @ and $\star$ are monotone.
$hyp(j@j)$
If $hyp(j)$, then $i * j \leq i$

Lemma 0.15 *1. If $\Gamma \leq \Gamma'$ and $\Gamma \vdash J$, then $\Gamma' \vdash J$.*

2. If $\vdash \Gamma$ ctx, then $\vdash \Gamma \star j$ ctx.

3. If $\Gamma * j, \Psi * j \vdash V \Leftarrow$ class and $\Gamma \vdash \sigma \Leftarrow \Psi$, then then $\Gamma@j \vdash V\{\sigma/\Psi\} \Leftarrow$ class.

Proof

1. Easy.

2. Suppose
$$\frac{\Gamma * j \vdash V \Leftarrow \text{class} \qquad \vdash \Gamma \text{ ctx}}{\vdash \Gamma, x :_j V \text{ ctx}}$$
We need
$$\frac{(\Gamma \star k) * j \star k \vdash V \Leftarrow \text{class} \qquad \vdash \Gamma \star k \text{ ctx}}{\vdash \Gamma \star k, x :_{j \star k} V \text{ ctx}}$$
So appeal to lemma and induction hypothesis.

3. Easy.

■

Abstract stories:
X tried to do something, and succeeded.
X tried to do something, and failed.
Once a thing happened that nobody expected.
X loved Y, but Y did not love X.
X loved Y, but Y did not know.
X loved Y, but Z, powerful, disapproved.
X had a secret, and Y discovered it.
X loved Y, and Z loved Y also, and so Z attacked X.
X did a terrible thing to Y, whom Z loved, and so Z attacked X.
X was angry at Y for no evident reason, confusing Y.
X built a thing, but Y destroyed it.
X was told a thing was impossible by Y, but X did it anyway.
X was told a thing was immoral by Y, but X did it anyway.

X looked for a thing, and found it.
X expected Y to do a thing, but Y did not do it.
X said to Y something that did not make sense to Z.
X believed a thing about Y which turned out to be false.
X attacked Y, and eventually X won.
X attacked Y, and eventually Y won.
X taught Y a skill, and Y used it against X.
X taught Y a skill, and Y used for X's benefit.
A thing that happens routinely, happened for the first time.
A thing that used to happen, happened for the last time.
A place was discovered.
X warned Y about a thing, and Y suffered for ignoring X's warning.
X predicted a thing, and it happened.
X tried to do a thing and succeeded in an unexpected way.

2007.7.26

ICFP was fun, but difficult. Plenty of secrets remaining.

Hacking on some wavelet-ish code. The idea is to first time-divide the signal f by partitioning it recursively into segments of equal

$$\int (f'(x))^2 \, dx$$

and then treat these as if they were equally long for the purpose of a Haar transform. The partition naturally results in a (generally unbalanced, because of the time-division bias) tree whose leaves are single samples, and doing the Haar is as simple as usual, by propagating sums and differences up the tree.

One can then chop up and rearrange the tree, or scale the Haar coefficients in some way dependent on their depth in the tree or whatever. Chopping out the top of the tree does a sort of weird high pass, and chopping out the bottom gives a very square-wavy low pass.

2007.7.27

I can find little merely type-theoretic evidence that the context restriction operator requires the stringency that proof-irrelevant *equality* does. I think the right setup is:

Specify a pointed partial order $(P, \leq, 0)$ and operations $\ominus, \oslash$. The former is for terms, the latter for contexts. We require only

$$x \ominus x \geq 0 \qquad x \oslash y \leq x \ominus y$$

and

$$\frac{x \leq y}{x \star m \leq y \star m} \qquad \frac{m \geq 0}{x \star m \leq x} \qquad (x \star z) * (y \star z) \geq x * y$$

for $\star, * \in \{\ominus, \oslash\}$. Some relevant rules:

$$\frac{x :_m \Pi\Psi.v \in \Gamma \qquad \Gamma \vdash \sigma \Leftarrow \Psi \qquad m \geq 0}{\Gamma \vdash x[\sigma] \Rightarrow v\{\sigma/\Psi\}}$$

$$\frac{\Gamma \ominus m \vdash M \Leftarrow A\{\sigma/\Psi\} \qquad \Gamma \vdash \sigma \Leftarrow \Psi}{\Gamma \vdash M.\sigma \Leftarrow \Psi, x :_m A}$$

Wait a second... it seems that $(x \ominus z) \oslash (y \ominus z) \geq x \oslash y$ leads to

$$(x \ominus y) \oslash (y \ominus y) \geq x \oslash y$$

and since $(y \ominus y) \geq 0$ we get

$$(x \ominus y) \oslash (y \ominus y) \leq x \ominus y$$

hence $x \oslash y \leq x \ominus y$, so that axiom is redundant. This means that if also always $x \oslash x \geq 0$, then the two operations are indistinguishable. In the case of proof irrelevance, $\div \oslash \div = \div \not\geq 0$.

2007.7.28

So in the case of the labelled linear system, I think I can conclude that the world variables are *not* proof-irrelevant hypotheses at all, since they are involved in types in such a way that their equational identity is critical. What *is* irrelevant are the proofs that *terms* belong to certain refinments.

Here is an alternative-to-HOAS idea:

```
* : type -> type.
@ : *A -> A -> type.
just : A -> *A.
yes : (just M) @ M.
+ : *A -> *A -> *A.
0 : *A.
inl : C @ N -> (C + D) @ N.
inr : D @ N -> (C + D) @ N.

ctx : type.
c : *ctx -> ctx.
sub : *ctx -> *ctx -> type.
tm : *ctx -> ctx -> type.
atm : *ctx -> type.

lam : tm G1 (c G2)
```

```
      <- atm (G1 + G2)

app : atm G1
      <- G1 @ (c G2)
      <- sub G1 G2.

nil : sub G1 0.
cons : sub G1 (G2 + G3)
       <- sub G1 G2
       <- sub G1 G3.

leaf : sub G (just A)
       <- tm G A.
```

or maybe

```
leaf : sub G (just (c A))
       <- (G + A) @ (c G')
       <- sub (G + A) G'.
```

which obviates the need for tm and atm.

Hmm... I don't really need this high-powered polymorphism at all, do I? I could just get by with a type `*ctx`.

Actually this bottoms out in deBruijn Hell anyway.

2007.7.29

Is it possible during unification to maintain the invariant that terms are well-typed but do not necessarily satisfy the label refinements?

2007.7.30

I think some equivalence like $\diamond A \equiv \Box(A \to p) \multimap p$ probably holds, analagous to Deepak's observation that $\bigcirc A \equiv (A \multimap p) \multimap p$ (which I think is in turn equivalent to $(A \to p) \multimap p$). The proof seems to depend on some focussing reasoning with the fresh atom p being negative.

2007.7.31

Consider unification with metavariables just floating around, but intrinsically contextually typed, like $u_{\Psi \vdash a}$. Variables u may appear in other us' types, but presume that there is some stratification to prevent circularity. A unification problem P is an unordered collection of equations $M \doteq M'$ (or $R \doteq R'$ or $S \doteq S'$...) and assignments $u \leftarrow R$. A problem P is *solvable* if it has at least one ground *instance*.

We speak of simply typed and fully typed instances according to whether substitutions for $u_{\Psi \vdash A}$ have to be merely simply typed or actually of type $\Psi \vdash A$ with all the dependencies correct. We are tacitly assuming everything in sight is at least simply well-typed. A simultaneous grounding substitution for all the free (unification) variables of a problem (and maybe for more variables that don't appear?) is an instance if it leaves all equalities true, and is a superset of all the assignments present in the problem. If a problem has an assignment in it that isn't (fully) well-typed, it has no typed instances.

A (not-necessarily-ground) substitution for some (maybe not all) $\vec{u}$ of the free variables of P is a $\vec{u}$-solution if, after carrying out the substitution, it is still solvable. Being solvable is obviously identical to having a $\cdot$-solution, namely id. Some observations, though: the set of $\vec{v}$ solutions is completely determined by the set of $\vec{u}$ solutions when $\vec{v} \subseteq \vec{u}$. The substitution θ is a $\vec{v}$-solution precisely when it can be extended to a $\vec{u}$-solution. This means that if P and P' share a set of $\vec{u}$-solutions, they also have the same set of $\vec{v}$-solutions.

We will impose on the design of the unification algorithm the constraint that if $P \mapsto P'$, then P and P' have the same set of $FV(P)$-solutions. If we make a further step from P' to P'', they will have the same set of $FV(P')$-solutions, but a fortiori, the same set of $FV(P)$ solutions, since the set of free variables monotonically increases.

Define $FV^*(P)$ to be the free variables of P *not* counting occurrences of variables on the left of $\leftarrow$, which *are* counted in $FV(P)$. We also maintain the invariant that if $u \leftarrow M \in P$, then $u \notin FV^*(P)$.

With all the above we should be able to show that the algorithm preserves sets of *simply* typed instances of unification problems, and so we can read off at the end what the solution is. If we care about dependent well-typedness, then maybe we can chew through the instantiation, making some other determinations about what must be equal for it to be well-typed.

However, it ought to be the case that unification also preserves typing. A unification problem P is well-typed if all of its equations and assignments are P-well-typed. An equation is P-well-typed if it *can* be given a context and a type such that, for any instance of P, both sides of the equation have the (substituted) type in the (substituted) context. An assignment is P well-typed if for any instance of P its right-hand side does have the (substituted) type in the (substituted) context, both drawn from the contextual type of the variable on the left.

There's a question here of whether I mean to quantify over *all* (simply-typed) instantiations or just the dependently well-typed ones. I think I can get away with the former, for it is a stronger thing to know once I'm finished with unification and need to read off the individual variable instantiations,

and I suspect it is still preserved as an invariant.

Inversion should look like

$$\frac{N[\sigma]^{-1} = N'}{P \wedge u[\sigma] \doteq N \mapsto P[N'/u] \wedge u \leftarrow N'}$$

No, wait, I need to think only about well-typed instantiations when defining well-typedness of unification problems: specifically, to be *able* to establish that the initial problem is well-typed.

2007.8.1

So unification should preserve sets of (simply-typed, from which follows typed) unifiers and preserve well-typedness.

Let us claim that the rule

$$\frac{R[\sigma]^{-1} = R'}{P \wedge u[\sigma] \doteq R \mapsto P[R'/u] \wedge u \leftarrow R'}$$

preserves unifiers for a set that *does* include u. That is, for any substitution θ containing R_s/u, we get the equivalence of $\models P\theta \wedge R_s[\sigma\theta] \doteq R\theta$ and $\models P[R'/u]\theta \wedge R_s \doteq R'\theta$. I guess I am supposing that R itself does not have any of the free variables of the simultaneous substitution θ. The latter expression breaks down anyhow into $\models P\theta[R'\theta/u] \wedge R_s \doteq R'\theta$ (again implicitly assuming u not free in the output of θ) which becomes just $\models P\theta \wedge R_s \doteq R'\theta$ because substituting twice for u has no effect.

Now for this step to go through it must be that σ is a pattern substitution, so hitting it with θ does nothing. We must determine that $R_s[\sigma] \doteq R\theta$ and $R_s \doteq R'\theta$ have the same solvability. We just need as a lemma that if $R[\sigma]^{-1} = R'$, then in fact $R = R'[\sigma]$. This means we are comparing $R_s[\sigma] \doteq R'\theta[\sigma]$ and $R_s \doteq R'\theta$.

The properties of pattern substitutions that make this true are that they commute with grounding metasubstitutions θ', and that they are injective.

Why does this step preserve typing? To suppose the antecedent is well-typed is to suppose that there is a Γ and a such that for all well-typed grounding substitutions θ that satisfy all the equations in P, we have

$$\Gamma\theta \vdash (u[\sigma])\theta \Rightarrow a\theta \qquad \Gamma\theta \vdash R\theta \Rightarrow a\theta$$

and need that

$$\Psi\theta \vdash R'\theta \Rightarrow b\theta$$

for $\Psi \vdash b$ being the type of $u \in \Delta$. We seem pretty stuck here!

Looking at Conal Elliott's thesis, it seems that he does resort to a strict partial order on unification *equations*, not just variables.

The problem I have understanding this idea is that even if I take the substitution that resolves only the equations antecedent to the current one, it still might instantiate u and leave me without explicit typing information about σ that I could pump through the definition of inversion and get something reasonable out. Could I maintain the invariant that the variables in an equation are disjoint from the ones required to be instantiated to account for its ill-typedness?

2007.8.2

Here's another unification idea. Define new typing judgments annotated by a subscript P so that $\Gamma \vdash_P M \Leftarrow A$ means 'M has type A modulo P'. Every particular rule remains exactly the same except for the synth-checking boundary where I say

$$\frac{\Gamma \vdash_P R \Rightarrow a \qquad a \equiv_P a'}{\Gamma \vdash_P R \Leftarrow a'}$$

where $a \equiv_P a'$ means: for all substitutions θ that leave P solvable, $a\theta = a'\theta$.

Now the invariant to be maintained is: (1) everything in sight is simply-typed. (2) If $M \doteq N$ is part of the unification problem P, then there is a Γ and A such that $\Gamma \vdash_P M \Leftarrow A$ and $\Gamma \vdash_P N \Leftarrow A$. (3) If $u \leftarrow R$ is part of the unification problem P, and $u : \Psi \vdash a \in \Delta$, then $\Psi \vdash_P R \Rightarrow a$.

The invariant for equations of synthesizing things is, I suppose, that they might synthesize to different things, but they will wind up equal.

To justify

$$\frac{R[\sigma]^{-1} = R'}{P \wedge u[\sigma] \doteq R \mapsto P[R'/u] \wedge u \leftarrow R'}$$

We get by assumption that

$$\frac{\Delta, u : \Psi \vdash a; \Gamma \vdash_{P'} \sigma \Leftarrow \Psi}{\Delta, u : \Psi \vdash a; \Gamma \vdash_{P'} u[\sigma] \Rightarrow a[\sigma]}$$

and

$$\Delta, u : \Psi \vdash a; \Gamma \vdash_{P'} R \Rightarrow a'$$

where $P' = P \wedge u[\sigma]$, and we are assuming $a[\sigma] \equiv_{P'} a'$. Now if R' is the *only* term that can possibly satisfy $u[\sigma] \doteq R$, and it should be by injectivity of pattern substitutions, then we also have, transferring from P' to P, (noting that u can't possibly occur in a or σ)

$$(\Delta; \Gamma \vdash_P \sigma \Leftarrow \Psi)[R'/u]$$

$$(\Delta; \Gamma \vdash_P R \Rightarrow a')[R'/u]$$
$$a[\sigma] \equiv_P a'[R'/u]$$

Now σ is a pattern substitution and R shouldn't contain u by the occurs-check, so we should be able to invoke some lemma like

Lemma 0.16 *Suppose* $\Delta; \Gamma \vdash_P \sigma \Leftarrow \Psi$ *and* $\Delta; \Gamma \vdash_P R \Rightarrow a'$. *If* $R[\sigma]^{-1} = R'$, *then* $\Delta; \Psi \vdash_P R' \Rightarrow a''$ *such that* $a''[\sigma] \equiv_P a'$.

to find that

$$(\Delta; \Psi)[R'/u] \vdash_P R' \Rightarrow a''$$

such that $a''[\sigma] \equiv_P a'[R'/u] \equiv_P a[\sigma]$. Again, pattern substitutions are injective, so $a'' \equiv_P a$, as required.

2007.8.3

Summary of things to mention to Frank:

- **Contextual stuff**. By restricting all variables to have contextual instead of functional arguments, both the substitution and identity theorems go through nicely.

  This tends to put some strong pressure on Πs and contexts to be *precisely* the same.

- **Base-type polymorphism stuff**. We can construe atomic types and atomic terms as belonging to the same syntactic category, likewise unifying type- and kind-level Π. Variables of classifier 'type' can occur in the context, obviating the need for signatures. They can be instantiated by base types only.

  This can be extended kinds, hyperkinds, etc., generally to a hierarchy of universes.

  This tends to put some pressure on types and objects to behave uniformly.

- **Multimodal stuff**.

  By reasoning abstractly about colon-decorators, we find that we need some operations $\ominus$ and $\oslash$ satisfying some usual monotonicity, anti-monotonicity, and compatibility axioms like

  $$\frac{x \leq y}{x \star m \leq y \star m} \qquad \frac{m \geq 0}{x \star m \leq x} \qquad (x \star z) * (y \star z) \geq x * y$$

  and also to get the identity property we need $x \ominus x \geq 0$ which entails $x \oslash y \leq x \ominus y$.

The usual multimodal stuff satisfies this with two extra modes $\bot$ and $\top$ to represent things removed from the context, and things promoted to be accessible by removed things. These two look exactly like proof irrelevance by themselves! Except even for judging context validity, we could take $\oslash = \ominus$ and allow irrelevant hypotheses to have types in the promoted context. This corroborates the idea that the proof-irrelevant modality (and variants of it) have an independent existence from the proof-irrelevant notion of equality.

- **Classical version of LF**.

  Using the contextual multi-modal approach, it is fairly easy to syntactically prove the correctness of both the double negation and modal translations back and forth between intuitionistic and classical logic. It seems that this should be extensible to dependent types more cleanly than the old labelled approach. The novelty that I missed before is not worrying about boxes as independent propositional operators, but only allowing them as 'parasites' on $\rightarrow$.

  Although it's not clear what the notion of well-formed type is in any proposal I can come up with, it seems like 'classical' LF wants to have a context of true or false assumptions where the basic type is some kind of Σ. However, true assumptions of Σ should just automatically decompose. The only *interesting* thing is assumptions that Σ is false, right? Could I get by with true and false hypotheses at base type, and false Σs? Seems unappealingly asymmetric.

  Wait, no, even the true Σs carry modality information that makes them interesting on the right. This is what makes classical modal logic not pervasively asynchronous, and precisely what makes it expressive enough to simulate intuitionistic logic.

- **Unification**. There doesn't seem to be any problem with 'circularity' in the typedness-justification of unification equations. Indeed the equation $u \doteq c\ u : p$ should be okay if $u : o$ and $c : o \rightarrow o$, precisely because it has no solutions.

  However, rigging up $\vdash_P$ seems kind of logical-relationsish, because I'm making what looks like a syntactic definition of something that nonetheless has real powerful universal quantifiers down at the $\equiv_P$ leaves.

  * * * * * *

  Circularity in the dependency graph among *variables*, however, seems really sketchy. It's not clear how complete pattern unification is,

anyway, so I'm not personally bothered by saying that $\exists w : o, u : a\ w \to o, v : a\ w.(w \doteq u\ v)$ can throw a 'constraints remaining' sort of error.

It looks like we want to extend the occurs-check for w down into the *type* of u and v somehow, but not in order to fail, just to postpone. If that check succeeds, it means that all the variables in the instantiation for w can be moved in Δ to the left of w.

2007.8.4

Actually, why not allow cyclic dependent types? When I write Ψ I can say to myself that what I really mean is something like $\Psi^s :: \Psi^d$ where $\square^s$ means 'take only the variables and their simple types' and $\square^d$ means 'take only the dependent types, as a list'. This way Ψ^d only refers back to its left.

The substitution principle is something like:
If $\Gamma \vdash J$ and $\Gamma \vdash \sigma \Leftarrow \Psi$ then $(\Gamma \setminus \Psi)\theta \vdash J\theta$.

$$
\begin{array}{rlll}
\text{Simple Contexts} & \Gamma & ::= & \cdot \mid (\Gamma, x : \Gamma) \\
\text{Contexts} & \Delta & ::= & \Gamma :: \Psi \\
\text{Type Lists} & \Psi & ::= & \cdot \mid \Psi, V \\
\text{Term Lists} & \sigma & ::= & \cdot \mid \sigma, M \\
\text{Types} & V & ::= & \Pi\Delta.v \\
\text{Terms} & M & ::= & \lambda\Gamma.R \\
\text{Base Types} & v & ::= & R \mid \text{type} \\
\text{Atomic Terms} & R & ::= & x[\sigma]
\end{array}
$$

$$\frac{\Delta + \Gamma :: \Psi \vdash R \Rightarrow v}{\Delta \vdash \lambda\Gamma.R \Leftarrow \Pi\Gamma :: \Psi.v}$$

$$\frac{x :: (\Pi\Gamma :: \Psi.v) \in \Delta \qquad \Delta \vdash \sigma \Leftarrow \Psi\{\sigma/\Gamma\}}{\Delta \vdash x[\sigma] \Rightarrow v\{\sigma/\Gamma\}}$$

$$\frac{\Delta \vdash M \Leftarrow V \qquad \Delta \vdash \sigma \Leftarrow \Psi}{\Delta \vdash \sigma, M \Leftarrow \Psi, V} \qquad \frac{}{\Delta \vdash \cdot \Leftarrow \cdot}$$

Simple typing:

$$\frac{\Gamma, \Gamma_t \vdash R \Rightarrow}{\Gamma \vdash \lambda\Gamma_t.R \Leftarrow \Gamma_t} \qquad \frac{x : \Gamma_s \in \Gamma \qquad \Gamma \vdash \sigma \Leftarrow \Gamma_s}{\Gamma \vdash x[\sigma] \Rightarrow}$$

$$\frac{\Gamma \vdash M \Leftarrow \Gamma_t \qquad \Gamma \vdash \sigma \Leftarrow \Gamma_s}{\Gamma \vdash \sigma, M \Leftarrow \Gamma_s, x : \Gamma_t} \qquad \frac{}{\Gamma \vdash \cdot \Leftarrow \cdot}$$

$$\frac{\Gamma \vdash \Psi \text{ ctx} \qquad \Gamma \vdash V \Leftarrow \text{class}}{\Gamma \vdash \Psi, V \text{ ctx}} \qquad \frac{}{\Gamma \vdash \cdot \text{ ctx}}$$

$$\frac{\Gamma, \Gamma' \vdash \Psi \text{ ctx} \qquad \Gamma, \Gamma' \vdash v \Rightarrow \text{class}}{\Gamma \vdash \Pi\Gamma' :: \Psi . v \Leftarrow \text{class}}$$

Context validity:

$$\frac{\Delta + \Gamma :: \Psi \vdash \Psi \text{ ctx} \qquad \Delta + \Gamma :: \Psi \vdash v \Rightarrow \text{class}}{\Delta \vdash \Pi\Gamma :: \Psi . v \Leftarrow \text{class}}$$

2007.8.5

Negative deBruijn indices don't seem to work the way I want them to with respect to these contexts, sadly. However they seem fine for expressing constants in the signature should I choose to have one, since they're invariant under shifts.

2007.8.6

An interesting property of the mean μ of some set of points $\bar{x} \in \mathbb{R}^n$ is that it is the x that minimizes the second moment about x. For set to zero the expression

$$\frac{d}{dx} \sum_i (x - x_i)^2 = \sum_i 2(x - x_i)$$

and you get

$$nx = \sum_i x = \sum_i \bar{x}_i$$

so $x = \frac{1}{n} \sum_i \bar{x}_i = \mu$.

This allows a generalization of mean and variance to distributions on graphs, metric spaces, simplicial complexes, etc. A point in a metric space is a mean if it minimizes the second moment about that point; the variance is the second moment about a mean.

2007.8.7

The ability to have circular dependencies seems to obviate even the need to have universes in the language. Just allow

$$\frac{\Gamma \vdash R \Rightarrow v}{\Gamma \vdash R \Rightarrow \text{class}}$$

and start your signature off with, essentially, type : type!

2007.8.8

Doing cut-elimination for the proof irrelevant modality by itself, it seems that an idempotent version of it is easier to show to be sound. The non-idempotent version still is, but it depends on the absence of decompositions at irrelevant modality, something that precisely the idempotent one allows; so that if they are mixed, (and share the same judgmental notion of $\div$) suddenly the non-idempotent is unsound. One might be able to get them in the same system by splitting $\div$ up into two judgments, one that permits decomposition under it, and one that doesn't.

2007.8.9

Noam pointed out to me Dummet's example of

$$\frac{A \vdash C \qquad B \vdash C}{\Gamma, A@B \vdash C}$$

as a connective that's sound only in the absence of other connectives like $\vee$. I wonder what its soundness proof looks like? I can't reconstruct it.

Even without completely focussing the system, I think introducing cyclic multi-Πs to LF would work fine. In fact, they would only really need to be used during abstraction for implicit arguments.

2007.8.10

Here is an attempt at a higher-order pattern unification algorithm. We take for granted the lack of ordering of unification equations and variables.

Postulate a dummy variable, call it $\__\tau$, at each simple type τ. It's actually the thing that pops out during pattern inversion when it's not in the image of the substitution. Consider it 'outside the PER' in the sense that it's not even considered equal to itself. The invariant on unification is that we seek well-typed closed things to put in evars, so we won't ever put _ in them, (at least not in closed terms) because it's also not well-typed.

Faced with

$$u[\sigma] \doteq M$$

where σ is a pattern, we reason like this:

First of all, if σ has any _s in it, (I guess I'm allowing _ into the pattern fragment, but I can weasel out of this by describing this as a move on substitutions that are all-but-patterns, except for precisely the presence of _) replace u with something that projects them out. We can do this, because in the absence of any non-pattern noise, the _ really would appear on the left and induce irreflexivity of the PER.

If M has u at the top with a pattern substitution, do intersection. If it's not a pattern on the right, postpone.

Now consider the occurrences of u on the right. If there are none, great, carry out inversion (creating _s) and execute the substitution throughout the rest of the unification problem. We must also effectively add the equation $M \doteq M$, but I expect this to be optimized away in most cases. If M doesn't have any occurrences of _, then we do indeed get to transform it into $\top$. Otherwise, we wait for some instantiation to bring it conclusively into (or out of, possibly? I suppose I could wind up with $\_ \doteq \_$ and have to fail) the PER.

If there's a rigid occurence of u somewhere, fail. If there's only flex occurrences, postpone.

That's it!

One thing that confuses me is it seems like Twelf should already have to cope with the flex-occurs-check postponement, even though the left-hand side is a pattern. Why doesn't this break the invariant of associating postponed equations with evars that have non-pattern substitutions?

* * * * * *

Yeah, considering both ways that $u[x] \doteq x \wedge u[\_] \doteq u[\_]$ could go depending on which equation was attacked first, we get $\_ \doteq \_$ in either case, correctly failing.

The other thing is, if I ever get _ in a rigid position during inversion, I pretty much know to fail. So it's really only an extension of the pattern language, (much like η-short variables) not of the term language.

2007.8.11

I think I could do cyclic types in an otherwise normal LF setting by having a pairwise cyclic Σ, like

$$\Sigma(x, y) : (A, B)$$

with rules like

$$\frac{\Gamma \vdash M_i \Leftarrow [M_1, M_2/x_1, x_2]A_i \qquad (\forall i)}{\Gamma \vdash \langle M_1, M_2 \rangle \Leftarrow \Sigma(x_1, x_2){:}(A_1, A_2)}$$

$$\frac{\Gamma, x_1 : A_1, x_2 : A_2 \vdash A_i : \text{type} \qquad (\forall i)}{\Gamma \vdash \Sigma(x_1, x_2){:}(A_1, A_2) : \text{type}}$$

$$\frac{\Gamma \vdash R \Rightarrow \Sigma(x_1, x_2){:}(A_1, A_2)}{\Gamma \vdash \pi_i R \Rightarrow [\pi_i R/x_i]A_i}$$

Some ideas for how to cope with a module system.

$$\begin{array}{rrcl}
\text{Contexts} & \Gamma & ::= & L \mid \{\bar{\ell} \triangleright \bar{x} : \bar{C}\} \mid \\
 & & & \Gamma \text{ where } x : V = M \mid \Gamma_1 \text{ and } \Gamma_2 \\
\text{Terms} & M & ::= & \lambda \bar{\ell} \triangleright \bar{x}.R \mid [\bar{\ell} = \bar{M}] \\
\text{Types} & V & ::= & \Pi \Psi.R \mid \Gamma \\
\text{Classifiers} & C & ::= & V \mid S_V(M) \mid S_{\text{ctx}}(\Gamma) \\
\text{Atomic Terms} & R & ::= & L[\bar{\ell} = \bar{M}] \\
\text{Long Identifiers} & L & ::= & x \mid L.\ell
\end{array}$$

2007.8.12

To do PCA on a dataset matrix $\vec{x}$ that has individual observed datapoints as rows (centered so the mean is zero) look at the eigenspace decomposition of the covariance matrix $x^\top x$. The eigenvectors are the principal components, and the eigenvalues are the variances.

2007.8.13

To reconcile labels with the style of unification I've been doing, it might be necessary to attach a label to each equation and maintain the typing invariant with respect to it.

A priority is to figure out which invariants I can actually get away with.

2007.8.14

Define a world-parameterized erasure $A/\!\!/p$ of HLF types into

$$\text{Simple Labelled Types} \quad \tau \quad ::= \quad p \mid \tau_1 \to \tau_2 \mid \forall \alpha.\tau$$

by

$$\begin{array}{rcl}
(\Pi x{:}A.B)/\!\!/p & = & (A/\!\!/\varepsilon) \to (B/\!\!/p) \\
(\forall \alpha.A)/\!\!/p & = & \forall \alpha.(A/\!\!/p) \\
(\downarrow \alpha.A)/\!\!/p & = & A[p/\alpha]/\!\!/p \\
(A@q)/\!\!/p & = & A/\!\!/q \\
(a \cdot S)/\!\!/p & = & p
\end{array}$$

It's pretty easy to define typing judgments for τ; I would conjecture the well-typed HLF terms are exactly the well-typed LF terms that satisfy this typing judgment also. This separation might make unification easier to talk about.

2007.8.15

I am leaning towards the equations themselves not being labelled, then; the unification problem proper exists in the LF-world, and the typing problems exist in simplified HLF. The way they communicate is through (term-only!) instantiations of existential variables, which have differing types (but

the same ultimately simplified) typs across the different parallel computations.

2007.8.16

So if I just create type-checking constraints at inversion time, they decompose sensibly until they get to evars, at which point we have more suspended typechecking constraints. The only way we should ever have some left over is if there are free term variables remaining — otherwise, all the unification of labels should come back either true or false.

Unless perhaps it's sensible (and/or required) to maintain connected constraints through multiple phases of logic program execution.

2007.8.17

It's still bothering me that I don't know what the *types* would be, really, in a classical version of LF. The point of double-negation translation is that all the types you really have access to are either $\neg\neg A^*$ or $\neg A^*$ for some translated A^*, but then that would seem to imply one level up that the only kinds you have access to are $\neg\neg K^*$ and $\neg K^*$, which I don't know how to make sense of. Maybe I have to go back and suppose that 'type' is really baked into the system somehow.

2007.8.18

In the event of working in the LF fragment of HLF, doing HLF unification should cause little to no performance penalty, since label-unification equation creation should take place in parallel with inversion, and the only equations that arise will be $\epsilon \doteq \epsilon$.

2007.8.19

The theorem corresponding to the definitions from the 14th is:

Theorem 0.17 $\Gamma \vdash_{HLF} J[p]$ *iff* $\Gamma^- \vdash_{LF} J^-$ *and* $\Gamma /\!\!/ \epsilon \vdash J /\!\!/ p$.

for some suitable notion of erasure $\square^-$ and proof system for types simplified with $/\!\!/$.

2007.8.20

Reading some work of Linger and Sheard. They provide at a certain point as a tempting non-example the signature

$$\begin{array}{rcl} nat & \div & \text{type} \\ + & : & nat \to nat \to nat \\ vec & \div & \Pi x \div nat.\text{type} \\ append & : & \Pi n, m \div nat.vec\ n \to vec\ m \to vec\ (n+m) \end{array}$$

This doesn't work because + wants its arguments to be relevant. I'm trying to see how I would do this in a refinement system. I would start out

just having vec : type, and then refining it to something like $vecn : nat \to$ type, but these are of different shapes. Does this make sense in william's system? I would also need two extra $\forall$s to cover the way that append is refined.

I keep coming back to the notion that $Ref(A)$ should be a kind if A is a type.

$$\frac{\Gamma \vdash A : \text{type}}{\Gamma \vdash Ref(A) : \text{kind}}$$

$$\frac{\Gamma \vdash r, s : Ref(A)}{\Gamma \vdash r \wedge s : Ref(A)}$$

$$\frac{}{\Gamma \vdash \top : Ref(A)}$$

$$\frac{\Gamma, x : B \vdash r : Ref(A)}{\Gamma \vdash \forall x{:}B.r : Ref(A)}$$

$$\frac{\Gamma \vdash r : Ref(A) \qquad \Gamma, x : A :: r \vdash s : Ref(B)}{\Gamma \vdash \Pi x{:}r.s : Ref(\Pi x{:}A.B)}$$

$$\frac{\Gamma \vdash R \Rightarrow Ref(a \cdot S)}{\Gamma \vdash R \Leftarrow Ref(a \cdot S)}$$

And then $vecn$ would be like $nat \to Ref(vec)$ and append would be

$$app : vec \to vec \to vec :: \forall n.\forall m.\forall d{:}plus\ n\ m\ p.vecn\ n \to vecn\ m \to vecn\ p$$

2007.8.21

A sketchy idea going back to some feelings I had about linear algebra and variable-for-variable substitution, connected also to 'first-class underscore' in unification:

Imagine that terms are applicative trees — without much loss, let's just say lists — of variables. The set of free variables of a term is kind of like the vector space that a vector lives in. E.g. consider a term that is abc. If we want to substitute a for b, we'll cons up a substitution that is the identity on a and c. For clarity, let's make the substitution totally change the world. We'll substitute A for b, A for a, and C for c. We want to make this substitution a linear function, and since the set of linear functions is itself a linear space, this should somehow be a sum of $[A/b]$, $[A/a]$, and $[C/c]$. Each of these should further break down into a 'recognition' covector and 'generation' vector. A covector for a transforms abc into the bitstring

(more generally field-element-string) 100 and then that 'scalar' times the vector A will be $A00$. The operation that is glueing together these thingies is *linear*, (not, for instance, bilinear) so that $A00 + 0A0 + 00B = AAB$.

I really don't know how to tie this in with higher-order terms, though. λ does funny things.

2007.8.22

Okay, so in unification each evar has a type and a context and a label. I want to believe that

$$\exists \overline{u :: \Gamma \vdash a[p]} \vdash \overline{M \doteq N}$$

is true iff

$$\exists \overline{u :: \Gamma^- \vdash a^-} \vdash \overline{M \doteq N}$$

and also

$$\overline{u :: \Gamma /\!\!/ \epsilon \vdash a /\!\!/ p}; \overline{\Gamma /\!\!/ \epsilon} \vdash \overline{u : a /\!\!/ p}$$

so at each evar down in the term I only have a substitution consisting of terms. The erasure $\Box^-$ actually knocks world variables out of the context. Really all I ought to do to show decidability of typing is to *first* show the split of typing into LF typing and world checking, and then describe residual unification over the latter.

2007.8.23

Raising is a bit sneaky. It's intriguingly unclear when I can do it. Consider the old equation

$$Z \hat{} x \doteq f \hat{} (X\ x) \hat{} (Y\ x)$$

for Z linear but X, Y not. I get out that $Z := f\ (X\ 1)\ (Y\ 1)$ and so it must be that $\alpha \vdash f\ (X\ 1)\ (Y\ 1) : \alpha$. As a consequence I'll make up two new world variables p, q depending on α and I'll find that $\alpha \vdash X\ 1 : p$ and $\alpha \vdash Y\ 1 : q$, and at the nil I'll get $\alpha \doteq pq$.

* * * * * *

Irritatingly, 'compiling' linearity into labels obscures some invariants that would I think be otherwise more evident in the original linear setting. It may be that I can express them and efficiently detect them in terms of labels, but it's not obvious yet that I can.

Some more coherent thoughts on linear algebra and the lambda calculus:

$$\begin{array}{rrcl} \text{Normal Terms} & M & ::= & R \mid \lambda x.M \\ \text{Atomic Terms} & R & ::= & kH \mid R\ N \\ \text{Heads} & H & ::= & c \mid x \mid 1 \end{array}$$

where k is some field element. 'Scalar terms' are those all of whose heads are 1, and regular terms never have 1 for heads. We can get a scalar term from a term and a variable by an operation $M(/x)$:

$$\begin{array}{rcl} (\lambda y.M)(/x) & = & \lambda y.(M(/x)) \\ (RN)(/x) & = & (R(/x))\ (N(/x)) \\ (kx)(/x) & = & k \\ (ky)(/x) & = & 0 \\ (kc)(/x) & = & 0 \end{array}$$

We can also take a scalar term and a head and make it into an ordinary term. We just go through and multiply all the heads by the given one. We write this as $M(x)$.

Now $M(/x)(y)$ is kind of like $[y/x]M$, except a lot of things go to zero. We need to define $M\mathbf{C}$, which preserves all the constant stuff.

$$\begin{array}{rcl} (\lambda y.M)\mathbf{C} & = & \lambda y.(M((/y)(y) + \mathbf{C})) \\ (RN)\mathbf{C} & = & (R\mathbf{C})\ (N\mathbf{C}) \\ (kx)\mathbf{C} & = & 0 \\ (kc)\mathbf{C} & = & c \end{array}$$

2007.8.24

Summary of things to mention to Frank:

- **Type Validity**. The issue of which atomic expressions should be well-formed is clear, and this is all I need for HLF metatheory to make novel sense. The question of which function types exist is still less clear.
- **Mutual Recursive Dependencies**. This seems to clear up the abstraction bug.
- **Term PER for Unification**. This seems like it might clear up the approximation-step bug.

2007.8.25

The Linger & Sheard thing is still gnawing at the fringes of my attention. The very fact that $\lambda x.x$ 'irrelevantly' has type $\Pi x \div o.o$ is highly peculiar,

yet it seems to hang together ok as a logic without worrying about erasure or equality or whatever.

This is yet a different system, isn't it, from the one where irrelevance is idempotent? It sure seems to be. I can't seem to prove $([[o] \to o] \to p) \to p$ in Awodey-Bauer.

2007.8.27

Game idea, with a competition structure akin to core wars, but a more cellular programming model: your task is to lay out little laser turrets and armor on, say, a 5 by 10 grid oriented vertically. The laser turrets each have some NESW orientation, and can also be armored. An armor cell is basically empty space plus some number of units of armor, and a gun can have the same armor added to it at the same cost. Say armor costs 1pt each unit, and adding the gun itself is an extra 1pt, but probably at least one unit of armor is mandatory for any occupied cell (though cells can also be left unoccupied) so a gun has a minimum cost of 2pts. Presumably there is some spending limit on points.

The player also specifies an order in which all the active things (in this case just the guns) execute. To compete, both players lay down their machines horizontally next to each other (with one flipped) and take turns executing the next gun in their queue. (If some get destroyed then later ones move up in the queue) If a gun fires at a player's own gun, the energy is captured and stored. The targetted gun will then fire a shot of 'more energy'. Each shot defaults to one energy unit, but received shots are added to that. If the energy of a shot equals or exceeds some bit of armor or enemy gun, then the target is destroyed, otherwise it fizzles. If the energy reaches the far side of the playing field, it scores points equal to the energy times some number that decreases with the height above the 'ground' at the bottom.

Turns are interleaved, but we can just average the two cases where each player has the initiative.

2007.8.28

Consider personality testing. Trying to come up with a nice abstract setting that reflects its measurement problems sufficiently. Suppose I have a set of questions Q and a set of persons P. I can imagine that the universe hands me a function $t : P \times Q \to \mathbb{R}$. Now if I have a distribution d over P (say, the set of people I am likely to encounter and interact with) I can start to talk about correlation of outputs of different questions. If I have q_1, q_2 then maybe it's meaningful to think about the covariance term $\sum_p d(p)t(p, q_1)t(p, q_2)$. But then again I probably should make sure that my data is mean-centered and varaiance-scaled, and so the distribution d plays an essential role in determining correlation.

The question I want to ask is something like: suppose Q is closed under linear combinations. (Hopefully the linear structure on Q I intend is obvious — construe it as the dual space of P with t acting like application) Can we discover any intrinsic structure in P without any guarantees that our set of questions might be wildly redundant?

2007.8.29

I find myself staring at the counterexample to Pfenning '01 that looks like

```
o : type. j,k : o.
a : o -> type.
b : {x:o} a x.
c : {x/o} {y/a x} o
c j (b j) =? c k (b k)
```

and thinking that it could be 'fixed' by a encoding into canonical irrelevant LF maybe by using sigmas to package up collections of arguments that are all irrelevant. The thing that threatens to break this is of course the possibility of interleaving irrelevant and relevant arguments.

2007.8.30

The damning thing about the above example — and I'm pretty sure I discovered this a long time ago — is that you can't replace an irrelevant subterm with another at the same type and retain well-typedness of the result, or even *typability* with respect to other irrelevant changes. Take the signature

```
o : type. j,k : o.
a : o -> type.
b : a j.
c : {x/o} {y/a x} o
```

and consider $c \circ j \circ b$. It's well-typed, but $c \circ k \circ b$ is not, nor is there anything we could replace b with to make it well-typed. Contrarily in a well-set-up system I should have a lemma like

Lemma 0.18 *If* $\Gamma, x \div A \vdash B : \text{type}$*, and* $\Gamma^{\oplus} \vdash M_1, M_2 \Leftarrow A$*, then for any* N *we have* $\Gamma \vdash N \Leftarrow [M_1/x]^A B$ *iff* $\Gamma \vdash N \Leftarrow [M_2/x]^A B$.

Indeed I might have

Lemma 0.19 *If* $B_1 \equiv_i B_2$*, then for any* N *we have* $\Gamma \vdash N \Leftarrow B_1$ *iff* $\Gamma \vdash N \Leftarrow B_2$.

Let me try to prove that. I have to generalize at least to

Lemma 0.20 *Suppose* $\Gamma_1 \equiv_i \Gamma_2$ *and* $A_1 \equiv_i A_2$.

- *If* $\Gamma_1 \vdash N \Leftarrow A_1$ *then* $\Gamma_2 \vdash N \Leftarrow A_2$.
- *If* $\Gamma_1 \vdash R \Rightarrow C$ *then there exists* $C' \equiv_i C$ *such that* $\Gamma_2 \vdash R \Rightarrow C'$.
- *If* $\Gamma_1 \vdash S : A_1 > C$ *then exists* $C' \equiv_i C$ *such that* $\Gamma_2 \vdash S : A_2 > C'$.

This probably requires some further well-typedness assumptions.

An attempt at the modal translation from intuitionistic multimodal LF into classical multimodal LF: A judgment $\Gamma \vdash M \Leftarrow V$ is taken to $\Gamma^\Box, k :_{0t} V^\Box \vdash M_k^\Box \Downarrow$.

$$\begin{array}{rcl}
(x :_n V)^\Box & = & x_{(n+1)f} : V^\Box \\
(\Pi\Psi.v)^\Box & = & \bigwedge(\Psi^\Box, k :_{0f} v^\Box) \\
(\lambda\hat{\Psi}.R)^\Box_{k_1} & = & k_1[\hat{\Psi}, k_2.R^\Box_{k_2}] \\
(x[\sigma])^\Box_k & = & x[(y.y \triangleright k).\sigma^\Box] \\
(M.\sigma)^\Box & = & (k.M^\Box_k).\sigma^\Box \\
R^\Box & = & t.R^\Box_t
\end{array}$$

The opposite translation takes $\Gamma \vdash E \Downarrow$ to $\Gamma^* \vdash E^* \Rightarrow \sharp$ and looks like

$$\begin{array}{rcl}
(x :_{nb} V)^\circ & = & x :_n \neg(V^*) \\
(x :_{nt} \bigwedge \Psi)^* & = & x :_n \neg\neg(\Psi^*) \\
(x :_{nf} \bigwedge \Psi)^* & = & x :_n \neg(\Psi^\circ) \\
(x :_{nt} v)^* & = & x :_n v^* \\
(x :_{nf} v)^* & = & x :_n \neg(v^*) \\
(t.E)^* & = & mktp[E^*] \\
(x \triangleright y)^* & = & y[x] \\
(x[\hat{\Psi}.E])^* & = & x[\lambda\hat{\Psi}.E^*] \\
(y.E)^* & = & \lambda y.E^*
\end{array}$$

where $\sharp$: type and $mktp$: $((\text{type} \to \sharp) \to \sharp) \to \text{type}$ are declared, and $\neg V = V \to \sharp$ and $\neg\Psi = \Pi\Psi.\sharp$.

2007.8.31

A person who recommends a certain lifestyle, or habit, or methodology: they had better follow it themselves, else they are a hypocrite. Yet they may still be criticized for recommending it, *merely* because it is that which they follow. But the truth may be: they are merely following it, because they have come to the belief that it is appropriate.

2007.9.1

I'm leaning back towards believing in the validity of the following sort of move in unification:

$$(X[\sigma] = Y[\cdots(c \cdot X[\sigma'])\cdots]) \mapsto (X \leftarrow Y[\cdots(\_)\cdots])$$

The general claim is that I can rewrite a term with a rigid head, which contains the variable being inverted in a rigid position, with underscore.

2007.9.2

Trying to make the thing from yesterday more formal. I think I need something like two mutually-recursive inversion stages, though it's possible it might be three. The top-level inversion, when it encounters a variable, will apply top-level inversion to each of that variables arguments, will it not? This is the notion that finding $c \cdot X$ while inverting for X should be replaced by _ even if it occurs deep within a term. If ordinary (top-level) inversion reaches a constant on the other hand, we get to go to *rigid* inversion perhaps.

Okay, so if ordinary inversion encounters the variable itself, we are stuck, *unless* we are at the very top-level, and both variables have pattern substitutions, in which case we can do intersection. If rigid inversion encounters the variable itself, it 'throws an exception' back up to the last place ordinary inversion was called, leaving an _ there. If rigid inversion encounters another variable, then it switches to regular inversion.

2007.9.3

$$\begin{array}{rcl}
u[\xi_1] \doteq u[\xi_2] & \mapsto & u \leftarrow v[\xi_1 \cap \xi_2] \\
u[\xi] \doteq M & \mapsto & u \leftarrow M[\xi]_f^{/u} \wedge M[\xi]_f^{/u}[\xi] \doteq M
\end{array}$$

$$\begin{array}{rcll}
(\lambda x.M)[\xi]_*^{/u} & = & \lambda x.(M[x/x.\xi]_*^{/u}) & \\
(x \cdot S)[\xi]_*^{/u} & = & y \cdot (S[\xi]_r^{/u}) & \text{if } x/y \in \xi \\
(x \cdot S)[\xi]_*^{/u} & = & _ & \text{if } x \notin \operatorname{cod} \xi \\
(c \cdot S)[\xi]_*^{/u} & = & c \cdot (S[\xi]_r^{/u}) & \\
(v[\sigma])[\xi]_*^{/u} & = & v[\sigma[\xi]_f^{/u}] & \text{if } v \neq u \\
(u[\sigma])[\xi]_r^{/u} & = & _ & \\
(u[\sigma])[\xi]_f^{/u} & = & fail &
\end{array}$$

2007.9.4

Ultimately the thing that justifies a move like

$$u[\xi_1] \doteq u[\xi_2] \mapsto u \leftarrow u'[\xi_1 \cap \xi_2]$$

is just the fact that $R[\xi_1] = R[\xi_2]$ iff there exists R' such that $R = R'[\xi_1 \cap \xi_2]$. All the business about the rest of the substitution vanishes because ξ_1, ξ_2 are patterns, and the remaining constraints in the unification problem go away

because either as the conclusion or assumption we have $R = R'[\xi_1 \cap \xi_2]$, which is the only difference between them.

Also I think we need only consider closed instantiations as the partial step before we talk about solvability of unification problems in the definition of $\bar{u}$-solutions. This simplifies things a lot.

2007.9.12

There's a very torsor-ish problem with news reporting: it's hard to genuinely say that a certain sort of story is over- or undereported, although it's easy to say that one source reports more of a certain kind than another.

2007.9.14

Type inference in HLF is subtler than I expected. Maybe I want to do it before η-expansion to allow type equality to guide my hand more?

2007.9.15

Wrote some code to visualize the set of words in a corpus, plotting log frequency against average position of the word in a sentence. I tried both taking an average of normalized word position (in the sense that 0 is beginning of a sentence and 1 is the end) and nonnormalized (just take the index of the word in the sentence, 0 means first word, 1 means second, etc.)

2007.9.16

Writing Bresenham's Algorithm in ML is relatively pleasant. I can interpose functions in front of the pixel-drawing routine to easily enforce monotonicity invariants that make the inner loop easier to write.

2007.9.21

Type inference in HLF is still quite tough. Doing it before η-expansion seems rather ugly. Three possibilities come to mind:

1. Allow for some extra term construct that signifies that η-expansion stops in a particular place.
2. Do type inference based on quantifying the variables that appear in the context or result.
3. Do type inference based on the worlds of the arguments.

The chief problem is that the quantification prefix of a free variable's type is not uniquely determined by the context in which it appears. Consider for instance

```
c : {p:w} ({b:w} {g:w} o @ (b * g)) -> o @ a -> type.
k : {a:w} c X X.
```

and also think about

```
c : (a -o D -> B -o E) -> type.
( = c : ({a,b:w} A @ a -> D -> B @ b -> E @ (a * b)) -> type.
k : c X.
```

Another notable issue is that apparently the lattice of refinements has a bottom, so why not just always choose that? (Partial answer: because then you would fail coverage. Fewer actual closed terms would actually satisfy the refinement, so fewer things would cover)

Something like

$$\bot(\tau_1 \to \cdots \tau_n \to \tau) = \forall \alpha_1, \ldots, \alpha_n, \alpha. \top^{\alpha_1}(\tau_1) \to \cdots \to \top^{\alpha_n}(\tau_n) \to \tau @ \alpha$$

$$\top^p(\tau_1 \to \cdots \tau_n \to \tau) = \bot(\tau_1) \to \cdots \to \bot(\tau_n) \to \tau @ p$$

Oh, but wait, we want (something analagous to) the most general unifier, not the least. So this is backwards. Nonetheless, I also have a top, which is erasing all the $\forall$s and setting everything to ϵ.

* * * * * *

I am mistaken about the $\top$; it's only so for closed terms. If you have world variables, it is not so. The definition of $\top^p$ illustrates that if you allow free world variables in the 'answer' then you do indeed again have a top. For all I know, all intersections also exist, but I doubt it.

More cases to meditate on:

```
c : {p:w} o @ p -> type.
k : {a:w} c X -> o @ a.
```

I think the right reconstruction for k involves an abstracted free variable:

```
k : {a:w}{b:w} {X:o @ b} c X -> o @ a.
```

(If we define subtyping by η-expansion, then a single η-expanded occurrence of a free variable actually does have as its most general type the type that it gets checked against, basically by definition.)

```
c : ({a,b:w} a -> a -> b) -> ({a,b:w} a -> b -> b) -> type.
k : c X X.
```

seems like X should reconstruct as `(a:w a -> a -> a)` so it seems some unification is going on.

2007.9.25

Frank pointed out that things get even hairier for doing refinement-intersection-by-unification if there are nested universal quantifiers. I ought

to investigate this. Apart from this issue, I fully expect that there are not always MGU-like universal things in the refinement lattice, precisely because there are not MGUs always in unification. A specific counterexample there would be nice.

2007.9.26

Got an antialiased polygon fill routine working okay in ML.

2007.10.2

Had a thought about the redundancy elimination stuff; could the gap between synth and checking be filled by *optional* type indices rather than type ascriptions?

2007.10.3

Our sense of history is ridiculously underdetermined. Without being able to have a conversation with the past, we claim inference of more about it that I would feel comfortable inferring *from* even a conversation.

2007.10.4

I like and feel like I ought to imitate Scott Aaronson's research statement. It's not merely a blurby 'I aim to do such-and-such' but a nice, thorough assessment of the open problems he cares about, why he cares about them, what they mean, what's been done about them, and what he hopes to do about them.

2007.10.6

In practice, the way I do logic risks approaching the study of arbitrarily recursively defined predicates, but some of them certainly seem more 'logical' in flavor. They are those that act as *consequence* relations, ones that establish a relation with a transitive and reflexive flavor (even if the relation is not actually binary), i.e. those that admit cut and identity. Moreover we expect a certain modularity from logical connectives, that each has its meaning explained *independently*. Sources of great worry:

1. Why can one get away with shoving things into the 'judgmental part' of thie logic?

2. How much of this can we fairly get away with?

3. What is the scope of the things we can fairly call 'judgmental'?

4. There is a similar tradeoff between the environment and the subject being in control over which OO-ish and type-theoretic FP design habits differ. The camp I'm in says: a piece of code should absolutely determine what it offers to the outside world, while the aspects phi-

losophy says that code should be available for manipulation by the environment.

2007.10.9

Watched a BBC video about Buddhism. A bit fluffy, but interesting. Again I am struck with the thought of: yes, it might be useful to practice mindfulness towards certain ends, but why accomplish *those* ends? I feel that Buddhism takes for granted the, ahem, desirability of ending suffering, and honestly, I'm quite willing to personally accept that, but this seems a kind of back-pedalling retcon as usual. What happens when we, hypothetically, eliminate the desire to eliminate desire?

2007.10.15

Digesting Linger's newer paper. He seems to have promotion properly sorted out now, but types are still divides-type in many places, and that seems very strange to me.

2007.10.16

Big questions:

1. What is the role of logic?
2. What is the role of the judgmental methodology?
3. What is the connection between that and category theory?
4. ...Multicategories?
5. What is the range of sensible judgmental notions?
6. Are judgments polar the same way that connectives are?

Some thoughts on a Stephen Pinker lecture on the modern decline of violence: modernity and technology exert a collectivizing force on individuals by providing social tools and systems that only make sense to or are only affordable by groups; roads, agricultural systems, intellectual and educational systems, economic systems. But they also exert an individuating force by making individuals' lives comfortable and longer. If individuals die frequently, they identify with their families out of necessity, but without that necessity, they are free to perpetuate their own ends.

2007.10.17

Consider the problem of economic incentives for work. It seems necessary to assign resources in exchange for work, with a concommitant guarantee of property rights thereafter in order to make good on the meaningfulness of that assignation. But to whatever degree that (possibly abstract)

assignation enables the actual control of physical resources, it is vulnerable to the 'problem' that physical resources are generally higher-order and can be used functionally to create more: nature has an interest rate. This creates a situation that has been considered unpleasant by many, that a person could survive *without* working, living off the fruit of machines.

This is not apparently so different from simple food collection ('hunter-gatherer' society minus hunting) except for the possibility that only some people are allowed the right to directly gather. Even if we reached a state of technological advancement where scarcity is not a problem even without labor (which I suppose we have not achieved yet, pace even those who suggest we have effectively solved the problem of scarcity *while including* labor, except that systematic polotical reasons create artificial scarcity still) something seems suboptimal if there is no incentive left to work to create wealth beyond necessity.

I was made to think about this because of Larry Lessig mentioning in a talk the thesis that copyright terms should only be extended prospectively, not retrospectively, because we do not need to give any further incentive for work already created. A slight strawmanning of this claim is that we might as well snatch away money from laborers five years after they have earned it, because, after all, they already *did* the work, and why should we further reward them for it now?

The counterargument to that seems to depend on the fact that the laborers contracted to do the work *only on the assumed condition* that they would own their pay in perpetuity or until they voluntarily decided to spend it. However, inflation is always a risk; the effective value of their wages in fact *can* be effectively 'stolen away' (a loaded phrase, though! I'd rather say 'may vanish') at any point in the future, in principle. This seems roughly analagous to the uncertainty the musician would face after their copyright terms expire; they may still be the beneficiary of the goodwill of the community and not have their work impolitely appropriated.

2007.10.18

Playing with a Twelf puzzle tom7 discovered stemming from unification involving peculiar higher-order 'specializing' clauses and functions such as

```
clause : pred ([x] A x) -> pred' (A k).
function : pred k -> outputtype -> type.
```

The solution to HO unification woes was to wedge in a propositional equality to postpone the unresolvable equations until another stage of splitting at which time the twelf-programmer is able to list possible cases of (effectively) a Huet imitation phase.

2007.10.19

Advisor meeting today left me feeling dissatisfied. I am somewhat stuck on a number of fronts.

2007.10.20

There ought to be a way of doing variable cases in Twelf that at least *appears* modular: if only one could make context definitions, abstract over them in higher-order lemma appeals, and incrementally add to them.

2007.10.21

I could sort of 'finitize' the polymorphism problems in HLF by doing the refinement language like this:

$$\begin{array}{rcl} \text{Unquantified Types } \upsilon & ::= & \tau \to \upsilon \mid p \\ \text{Quantified Types } \tau & ::= & \forall \alpha.\upsilon \\ \text{Worlds } p & ::= & \epsilon \mid p * (\alpha, n) \\ \text{Substitutions } \theta & ::= & \epsilon \mid \theta, [p/n] \end{array}$$

With typing rules like

$$\frac{\Gamma, \alpha : \mathsf{w} \vdash N \Leftarrow \upsilon}{\Gamma \vdash N \Leftarrow \forall \alpha.\upsilon}$$

$$\frac{\Gamma, x : \tau \vdash N \Leftarrow \upsilon}{\Gamma \vdash \lambda x.N \Leftarrow \tau \to \upsilon}$$

$$\frac{x : \forall \alpha.\upsilon \in \Gamma \qquad \Gamma \vdash \theta : \overline{\mathsf{w}} \qquad \Gamma \vdash S : \tau\{\theta/\alpha\} > p}{\Gamma \vdash x \cdot S \Leftarrow p}$$

$$\frac{}{\Gamma \vdash () : p > p}$$

$$\frac{\Gamma \vdash M \Leftarrow \tau \qquad \Gamma \vdash S : \upsilon > p}{\Gamma \vdash (M; S) : \tau \to \upsilon > p}$$

2007.10.22

Semiunification is when you have inequalities instead of equalities — seems to get undecidable even faster than unification.

2007.10.23

I worry a bit still about substitution principles as concerns chrisamaphone's attempt at encoding OLF with operators rather than two species of worlds.

2007.10.24

Whoops, the inequalities in semiunification are *not* held abstract, as I suspected, but refer to the partial order of instantiation. Reading:

A Larger Decidable Semiunification Problem, Brad Lushman and Gordon V. Cormack.

Polymorphic Type Inference and Semi-Unification, Fritz Henglein's PhD thesis.

2007.10.25

Some discussion today with tom7 regarding the meaning of values, and dan licata about binding.

A short play:

[There is table, with a cheeseburger on it. X stands in front of it, with his arms crossed behind his back, looking attentive. Y saunters in. Y sees the cheeseburger, delighted, and reaches for it. X, at the last minute, observes what is going on, and smacks Y's hand out of the way]

Y: You can't do that!

X: Why?

Y: That's the very last cheeseburger in the world!

X: Oh. I see. I'll — I'll have the salad, then.

Y: Well — actually that's the last piece of any kind of food at all in the world. Happens it's a cheeseburger.

X: So, then.

Y: *[Pedagogically]* Mustn't eat it.

[A little time passes. X visibly impatient.]

X: Er — Why not?

Y: Obviously, if you eat it, there'll be none left!

X: What good is it, exactly, to save it, if nobody gets to eat it?

Y: Well, you — you see the implications of — of — *[Seems to be doing arithmetic on his fingers]* future generations will — uh — *[Looks at X. Their eyes lock, as if at a duel]*

[Both lunge at the cheeseburger, wrestle over it. In the end Y gets it.]

Y: *[mid-chew]* The problem is, now I want fries.

2007.10.26

For a while I thought the natural unification-based reconstruction of

```
b : type.
c : ((b -o b) -o b) -> type.
- : c D.
```

which is to say, assigning to D something like the refinement

$$\forall\gamma : \mathsf{w} \to \mathsf{w} \to \mathsf{w}.(\forall\alpha.(\forall\beta.\gamma[\alpha,\beta] \to \gamma[\alpha,\beta] * \alpha) \to \alpha)$$

was totally wrong. But really, instantiating γ with a projection that picks out β loses no generality precisely because of the nested positive occurrence of the quantifier binding β. If I can figure out what kind of move justifies this rewriting (notably back into a fragment I know how to do abstraction over!) then I might have a decent reconstruction algorithm, and one which is not too tied to LLF.

2007.10.27

We say $\tau_1 \leq \tau_2$ when they have the same shape (up to $\to$ and $\forall$) and when there exists a substitution θ over the free world variables of τ_2 (which may have in its codomain free variables of τ_1) such that

$$x : \tau_1 \vdash \eta_{\tau_1}(x) \Leftarrow \tau_2$$

where η-expansion is defined, as expected, by

$$\begin{array}{rcl} \eta_{\tau_1 \to \tau_2}(R) & := & \lambda x.\eta_{\tau_2}(R\ (\eta_{\tau_1}(x))) \\ \eta_{\forall\tau}(R) & := & \Lambda(\eta_\tau(R\ \_)) \\ \eta_a(R) & := & R \end{array}$$

Lemma 0.21 *If all the occurrences of some positively bound variable β are in the local contexts of one free variable, then we may replace that free variable with β without loss.*

Proof Sketch There are two directions to show. The first,

$$(\forall\beta.\tau_1) \to \tau_2 \leq (\forall\beta.\tau_1\{P[\beta]/\beta\}) \to \tau_2$$

is trivial by instantiation. The second,

$$(\forall\beta.\tau_1\{P[\beta]/\beta\}) \to \tau_2 \leq (\forall\beta.\tau_1) \to \tau_2$$

proceeds by seeing that

$$\dfrac{\dfrac{\dfrac{\dfrac{\tau_1\{P[\beta]/\beta\} = \tau_1\{P[\beta]/\beta\}}{\beta : \mathsf{w} \vdash y\ \_ \Leftarrow \tau_1\{P[\beta]/\beta\}}}{\vdash \Lambda(y\ \_) \Leftarrow \forall\beta.\tau_1\{P[\beta]/\beta\}}}{y : \forall\beta.\tau_1 \vdash x\ (\Lambda(y\ \_)) \Leftarrow \tau_2}}{x : (\forall\beta.\tau_1\{P[\beta]/\beta\}) \to \tau_2 \vdash \lambda y.x\ (\Lambda(y\ \_)) \Leftarrow (\forall\beta.\tau_1) \to \tau_2}$$

■

Actually, I think I want to reason like this: My goal is to show that if a type is of the form $C[\forall\beta.\tau]$, where the context-hole is in negative position, then

$$C[\forall\beta.\tau] \equiv C[\forall\beta.(\tau\{P[\vec{p},\beta]/\beta\})]$$

where P is a free variable not already appearing in C or τ, and where $\vec{p}$ is anything else that might be formed from stuff currently in context. The easy direction is

$$C[\forall\beta.\tau] \leq C[\forall\beta.(\tau\{P[\vec{p},\beta]/\beta\})]$$

which I get by instantiation. The less trivial but still reasonable direction is the lemma

Lemma 0.22

1. *If C is a negative context, $C[\forall\beta.\tau] \geq C[\forall\beta.(\tau\{p/\beta\})]$.*
2. *If C is a positive context, $C[\forall\beta.\tau] \leq C[\forall\beta.(\tau\{p/\beta\})]$.*

2007.10.28

Fragments for a play about identity:

[M1 and M2 are lying in bed]

M1: *[Contentedly, but not looking directly at her]* Mmm, Mary. Last night —

M2: *[A light bulb turning on]* My name is Michael!

M1: Uh, *my* name's Michael.

M2: Yes!

M1: Right.

M2: Yes, exactly. My name's Michael.

M1: No.

M2: *[Confused]* You just said yes a second ago.

[...]

M1: You can't — you can't just subsume your whole self into mine! I thought — you told me you were a feminist.

M2: I *am* a feminist. I believe in *rattles off some standard stuff. [Conspiratorially]* Plus it helps me pick up chicks.

M1: You're a lesbian?

M2: *[Laughing]* No! Of course not! I'm a perfectly ordinary heterosexual...

[M1 looks relieved]

M2: ...male.

[M1 immediately snaps back to not so relieved. He lifts up the sheet, to inspect those parts of M2 hidden from the audience. He smiles again.]

M1: No, you're not.

[M2 does the same in reverse.]
M2: *[Inexplicably sultry]* Yes, I am.
[...]
M1: I think you mean 'you'.
M2: I think you mean '*me*'.
M1: I think *I* mean '*me*'.

2007.10.29

I think I should be able to prove the same sort of polarity-sensitive two-part lemma to show that refinement reconstruction can actually assign the (a?) right type to the eta expansion of a variable with unknown full refinement but at least known simple refinement.

2007.10.30

The thing to remember about mutual recursion and parser combinators is that the unit-applying auxiliary function that usually gets called $ actually needs to be part of the combinator library, and applies its argument to unit 'inside' the application to the further argument that is actually the next bit of stuff coming off the stream.

2007.10.31

It *seems* at least to be important to have a fake sort of binder (to correspond to universal quantification over worlds) in the syntax, which actually increments deBruijn indices appropriately, so that when we toss a variable into the context upon decomposing a $\forall$, we don't have to do crazy shifting nonsense on the term side.

* * * * * *

So here's the plan about type reconstruction. Shapes are given by

$$\begin{array}{rcl} \text{Shapes } s & ::= & \forall s \mid s \to s \mid \bullet \\ \text{Refinements } \tau & ::= & \forall \alpha.\tau \mid \tau \to \tau \mid p \end{array}$$

And given such a thing we can create a evar-laden guess as to what the type of a Π-bound but un-type-ascribed variable is:

$$\frac{\Gamma \vdash g(s_1) = \tau_1 \qquad \Gamma \vdash g(s_2) = \tau_2}{\Gamma \vdash g(s_1 \to s_2) = \tau_1 \to \tau_2} \qquad \frac{}{\Gamma \vdash g(\bullet) = P[\hat{\Gamma}]}$$

$$\frac{\Gamma, \alpha : \mathsf{w} \vdash g(s) = \tau}{\Gamma \vdash g(\forall s) = \forall \alpha.\tau}$$

The system for collecting unification constraints from type checking is easy:

$$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2/C}{\Gamma \vdash \lambda x.M : \tau_1 \to \tau_2/C} \qquad \frac{\Gamma \vdash M : \tau_1/C_1 \qquad \Gamma \vdash S : \tau_2 > p/C_2}{\Gamma \vdash (M; S) : \tau_1 \to \tau_2/C_1 \wedge C_2}$$

$$\frac{\Gamma, \alpha : \mathsf{w} \vdash M : \tau/C}{\Gamma \vdash \Lambda M : \forall \alpha.\tau/C} \qquad \frac{\Gamma \vdash S : \tau\{P[\mathsf{w}(\Gamma)]/\alpha\} > p/C}{\Gamma \vdash (\_; S) : \forall \alpha.\tau > p/C}$$

$$\frac{x : \tau \in \Gamma \qquad \Gamma \vdash S : \tau > p/C}{\Gamma \vdash x \cdot S : p/C} \qquad \frac{}{\Gamma \vdash () : q > p/p \doteq q}$$

Now η-expansion can be defined by shape:

$$\begin{array}{rcl} \eta_{s_1 \to s_2}(R) & := & \lambda x.\eta_{s_2}(R\ \eta_{s_1}(x)) \\ \eta_{\forall s}(R) & := & \Lambda \eta_s(R\ \_) \\ \eta_\bullet(R) & := & R \end{array}$$

The claim is something like

Lemma 0.23 *Let s be the shape of τ. Let τ' be one particular choice of evars for $g(\tau)$. Run type inference*

$$x : \tau' \vdash \eta_s(x) : \tau/C$$

Then doing unification on C will result in a substitution that will make τ' equivalent to τ.

The trouble is that this isn't quite true — it takes some extra moves in the abstraction phase to eliminate the remaining free variables in favor of bound variables of the appropriate polarity.

2007.11.1

What would a constructive theory of probability look like? Neel suggested to me the axioms

$$P(A) = 1 \text{ if } \vdash A$$

$$P(A) = 0 \text{ if } \vdash \neg A$$

$$P(A) \leq P(B) \text{ if } A \vdash B$$

$$P(A \vee B) + P(A \wedge B) = P(A) + P(B)$$

And I think I could simplify this to just

$$P(\top) = 1$$

$$P(\bot) = 0$$

$$P(A) \leq P(B) \text{ if } A \vdash B$$

$$P(A \vee B) + P(A \wedge B) = P(A) + P(B)$$

Then how do we account for conditioning? Maybe by changing them to

$$P_\Gamma(\top) = 1$$

$$P_\cdot(\bot) = 0$$

$$P_\Gamma(A) \leq P_\Gamma(B) \text{ if } \Gamma, A \vdash B$$

$$P_\Gamma(A \vee B) + P_\Gamma(A \wedge B) = P_\Gamma(A) + P_\Gamma(B)$$

$$P_\Gamma(A|B) = P_{\Gamma,B}(A)$$

$$P_\Gamma(A|B)P_\Gamma(B) = P_\Gamma(A \wedge B)$$

2007.11.2

Here is a notion for how to use substructural features to encode freshness and apartness of names, following bob's suggestion. The setup is quite similar to the encoding of a stack machine for miniml in LLF done by iliano and frank.

```
name : type.
val : type.
inst : type.
final : type.

% stores are name/val pair lists
store : type.
stnil : store.
stcons : name -0 val -> store -> store.

% some expressions and values
ref  : exp -> exp.
deref : exp -> exp.
loc : name -0 val.

% some instructions
ev : exp -> inst. % evaluate
return : val -> inst.
ref1 : val -> inst. % suspended ref
deref1 : val -> inst. % suspended deref
```

```
% this is the same hack as in frank and iliano's thing,
% to allow final states to be open
new* : (name -0 final) -> final.

% continuations are (val -> inst) lists
cont : type.
init : cont.
;    : cont -> (val -> inst) -> cont.

% exec taks a store as well
exec  : store -> cont -> inst -> final -> type.

% these are the easy ones that just push
% something onto the stack
ex_ref  : exec ST K (ev (ref E)) W
   o- exec ST (K ; ref1) (ev E) W.

ex_deref  : exec ST K (ev (deref E)) W
   o- exec ST (K ; deref1) (ev E) W.

% these are when we get back a value
ex_ref1 : exec ST K (ref1 V) (new W)
   o- ({n :^ name}
  exec (stcons n V ST) K (return (loc n)) (W n)).
ex_deref1 : exec ST K (deref1 (loc N)) W
   o- (lookup ST N V & exec ST K (return V) W).

% lookup a name in the store
lookup : store -0 name -0 value -> type.

lookup/here : !a (lookup (stcons N V S) N V).
lookup/there : lookup (stcons N' _ S) N' V
               o- (N # N' & lookup S N V).

% apartness
# : name -0 name -0 type.
irrefl: {l :^ loc} {l' :^ loc} !a (l # l').
```

The new notations are `{x :^ A}` which means $\forall\alpha : \mathsf{w}.\Pi x : A@\alpha$, an affine modality `!a(A)` which means $\downarrow\beta.\forall\alpha.A@(\alpha * \beta)$, and zero-use implication `A -0 B` which means $\forall\alpha.(A@\alpha) \multimap B$.

2007.11.3

The reason that nullary implication works on kinds is precisely that it has *no* funny business that it applies to the codomain.

2007.11.4

Here is a problem with type reconstruction still:

$$x : \forall\alpha.\forall\beta.P[\alpha,\beta] \vdash \Lambda\Lambda(x\ \_\ \_) \Leftarrow \forall\alpha.\forall\beta.p(\alpha,\beta)$$

$$\alpha : \mathsf{w}, \beta : \mathsf{w} \vdash x\ \_\ \_ \Leftarrow p(\alpha,\beta)$$

$$P[Q[\alpha,\beta], R[\alpha,\beta]] = p(\alpha,\beta)$$

like if $p(\alpha,\beta) = \alpha * \alpha * \beta$ then $P = \alpha * \beta$ and $R = \alpha * \beta$ and $Q = \alpha$ works, and so does $P = \alpha * \alpha * \beta$ and $Q = \alpha$ and $R = \beta$. Even in like

$$x : \forall\alpha.P[\alpha] \vdash \Lambda(x\ \_) \Leftarrow \forall\alpha.\alpha * \alpha$$

we have an ambiguity in the equation $P[Q[\alpha]] = \alpha * \alpha$ as to whether P or Q duplicates its argument.

2007.11.5

The above problem seems to be benign for linear (and n-ary) functions precisely because there is the single α attached to the domain that makes appropriate-polarity equations still unambiguous.

During my advisor meeting Frank drew my attention to the question of whether evars created during coverage checking need to have underscores attached to them. I don't believe they do.

2007.11.6

Need to figure out whether Alberto and Frank's 0-use business is isomorphic to what I'm doing.

2007.11.7

Ok, so the nullary arrow in either system is

$$\frac{\Gamma, \alpha : \mathsf{w}, x : A@\alpha \vdash M \Leftarrow B[p]}{\Gamma \vdash \hat{\lambda}x.M \Leftarrow A \not\multimap B[p]}$$

and their intro rule I can I can consider the same, but their elim is something like

(mine)

$$\frac{\Gamma \vdash R \Rightarrow A \not\multimap B[p] \qquad \Gamma@\epsilon \vdash N \Leftarrow A[\epsilon]}{\Gamma \vdash R\hat{}N \Rightarrow B[p]}$$

where '$\Gamma@\epsilon$' is an operation that I can't necessarily see how to define; perhaps grab all world variables in Γ and substitute ϵ for them? No, this doesn't seem to satisfy local contraction.

Is there any left rule that would go with a modality that on the right does the following?

$$\frac{\Gamma@\epsilon \vdash A[\epsilon]}{\Gamma \vdash \star A[p]}$$

2007.11.8

The nullary arrows are definitely different. The promotion in the Momigliano-Pfenning system allows irrelevant things (once promoted) to be used as the arguments of unrestricted functions, which isn't the case in HLF unless unrestricted functions are interpreted as ω-ary use.

2007.11.9

Some funny things go on algebraically in chrisamaphone's project. Distributivity isn't obvious at all.

2007.11.10

In fact, is there special behavior for when we try to weak-bang a *singleton* ordered context? Not sure.

2007.11.11

The ability for some ASL verbs to incorporate subject and object actually depends on their gestural structure — I suppose this is like a language that marks some feature by voicing, an operation only supported, let us suppose, by some consonants in the phonetic inventory of the language. Like, it might have m and n and lack voiceless counterparts of them. (Voiceless nasals are pretty rare, right?)

2007.11.12

I am worried still that any paper I might write about unification in HLF would wind up not essentially being about unification since the separation is so straightforward. Nonetheless, the term constructs that I added to make type reconstruction easier complicate things a tiny bit.

2007.11.13

Talked with neel about a strategy for proving the correctness of stateful but 'essentially functional' things like gensym.

2007.11.14

As for unification:

I think I have paged back in the argument for why preserving all sets

of unifiers preserves well-typed unifiers. I expect the definition of $\equiv_P$ is concerned with equality under only simply-typed solutions to P.

The 'underscore' approach might require underscores to appear in general head positions, not just substitutions, if we push inversion across lambdas. In this case the notion of 'rigid occurrence' needs to avoid locally bound variables.

Thing to watch out for: what guarantees that types are all sensible when I project out just one underscored argument in a pattern?

The right way, I think, to phrase the notion of solutions in a given set of variables, is in terms of projected subsets of simultaneous substitutions for all variables.

Mental health self-instruction: no more craigslist. No more reddit, digg, or del.icio.us frontpage.

2007.11.15

Talked to rob a bit about logic programming and Girard and stuff. I still can't wrap my head around *why* Girard thinks the way he does about logic. For my own part, the biggest blot is a lack of understanding of what cut principles are okay.

2007.11.21

A new thought on the completeness of focussing.

Translate into ordered logic with two positive atoms p and q that act as key-tokens.

$$A = vF^+ \mid vF^-$$
$$F^+ = F^+ \otimes F^+ \mid d^+A$$
$$F^- = F^+ \multimap F^- \mid d^-A$$

Define $\overleftarrow{X}$ and $\overleftarrow{\overline{A}}$ and $\overrightarrow{X}$ and $\overrightarrow{\overline{A}}$:

$$\overleftarrow{\overline{A}} = p \twoheadrightarrow \overleftarrow{A}$$
$$\overrightarrow{\overline{A}} = p \bullet \overrightarrow{A}$$

X	$\overleftarrow{X}$	$\overrightarrow{X}$
vF^+	$(q \twoheadrightarrow p) \bullet \overleftarrow{F^+} \bullet q$	$\overrightarrow{F^+}$
vF^-	$\overleftarrow{F^-}$	$(q \twoheadrightarrow p) \twoheadrightarrow \overrightarrow{F^-}$
d^+A	$q \twoheadrightarrow (q \bullet ! \overleftarrow{\overline{A}})$	$p \twoheadrightarrow \overrightarrow{\overline{A}}$
d^-A	$p \bullet ! \overleftarrow{\overline{A}}$	$q \twoheadrightarrow \overrightarrow{\overline{A}}$
$F_1^+ \otimes F_2^+$	$q \twoheadrightarrow (\overleftarrow{F_1^+} \bullet \overleftarrow{F_1^+} \bullet q)$	$\overrightarrow{F_1^+} \bullet \overrightarrow{F_2^+}$
$F^+ \multimap F^-$	$\overrightarrow{F^+} \twoheadrightarrow \overleftarrow{F^-}$	$\overleftarrow{F^+} \twoheadrightarrow \overrightarrow{F_-}$

Now prove:

Lemma 0.24

1. $\overleftarrow{\overrightarrow{A}}; p \vdash \overrightarrow{\overrightarrow{A}}$
2. *If* $\Omega, q, \overrightarrow{F^+}, \Omega' \vdash C$, *then* $\Omega, \overleftarrow{F^+}, q, \Omega' \vdash C$
3. *If* $\Omega, q \vdash \overleftarrow{F^-}$, *then* $\Omega \vdash \overrightarrow{F^-}$

Proof By induction on the proposition.

1.

Case: vF^+.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash p} \quad \dfrac{}{\overrightarrow{F^+} \vdash \overrightarrow{F^+}}\, id}{p, \overrightarrow{F^+} \vdash p \bullet \overrightarrow{F^+}} \bullet R \quad q \vdash q}{(q \twoheadrightarrow p), q, \overrightarrow{F^+} \vdash p \bullet \overrightarrow{F^+}} \twoheadrightarrow L}{(q \twoheadrightarrow p), \overleftarrow{F^+}, q \vdash p \bullet \overrightarrow{F^+}}\, i.h.}{(q \twoheadrightarrow p) \bullet \overleftarrow{F^+} \bullet q \vdash p \bullet \overrightarrow{F^+}} \bullet L \quad p \vdash p}{p \twoheadrightarrow ((q \twoheadrightarrow p) \bullet \overleftarrow{F^+} \bullet q); p \vdash p \bullet \overrightarrow{F^+}} \twoheadrightarrow L$$

Case: vF^-.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overleftarrow{F^-} \vdash \overleftarrow{F^-}}\, id \quad \dfrac{}{p \vdash p}}{p \twoheadrightarrow \overleftarrow{F^-}; p \vdash \overleftarrow{F^-}} \twoheadrightarrow L \quad q \vdash q}{p \twoheadrightarrow \overleftarrow{F^-}; q \twoheadrightarrow p, q \vdash \overleftarrow{F^-}} \twoheadrightarrow L}{p \twoheadrightarrow \overleftarrow{F^-}; q \twoheadrightarrow p \vdash \overrightarrow{F^-}}\, i.h.}{p \twoheadrightarrow \overleftarrow{F^-}; \cdot \vdash (q \twoheadrightarrow p) \twoheadrightarrow \overrightarrow{F^-}} \twoheadrightarrow L \quad p \vdash p}{p \twoheadrightarrow \overleftarrow{F^-}; p \vdash p \bullet ((q \twoheadrightarrow p) \twoheadrightarrow \overrightarrow{F^-})} \bullet R$$

2.

Case: d^+A.

$$\dfrac{\dfrac{\dfrac{}{\overleftarrow{\overrightarrow{A}}; p \vdash \overrightarrow{\overrightarrow{A}}}\, i.h.}{!\overleftarrow{\overrightarrow{A}} \vdash p \twoheadrightarrow \overrightarrow{\overrightarrow{A}}}\, !L, \twoheadrightarrow R \qquad \Omega, q, p \twoheadrightarrow \overrightarrow{\overrightarrow{A}}, \Omega' \vdash C}{\dfrac{\dfrac{\Omega, q, !\overleftarrow{\overrightarrow{A}}, \Omega' \vdash C}{\Omega, q \bullet !\overleftarrow{\overrightarrow{A}}, \Omega' \vdash C}\, \bullet L \qquad q \vdash q}{\Omega, q \twoheadrightarrow (q \bullet !\overleftarrow{\overrightarrow{A}}), q, \Omega' \vdash C}\, \twoheadrightarrow L}\, cut}$$

Case: $F_1^+ \otimes F_2^+$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overrightarrow{\overrightarrow{F_1^+}} \vdash \overrightarrow{\overrightarrow{F_1^+}}}\, id \quad \dfrac{}{\overrightarrow{\overrightarrow{F_2^+}} \vdash \overrightarrow{\overrightarrow{F_2^+}}}\, id}{\overrightarrow{F_1^+}, \overrightarrow{F_2^+} \vdash \overrightarrow{F_1^+} \bullet \overrightarrow{F_2^+}}\, \bullet R \qquad \Omega, q, \overrightarrow{F_1^+} \bullet \overrightarrow{F_2^+}, \Omega' \vdash C}{\dfrac{\dfrac{\dfrac{\dfrac{\Omega, q, \overrightarrow{F_1^+}, \overrightarrow{F_2^+}, \Omega' \vdash C}{\Omega, \overleftarrow{F_1^+}, q, \overrightarrow{F_2^+}, \Omega' \vdash C}\, i.h.}{\Omega, \overleftarrow{F_1^+}, \overleftarrow{F_2^+}, q, \Omega' \vdash C}\, i.h.}{\Omega, \overleftarrow{F_1^+} \bullet \overleftarrow{F_2^+} \bullet q, \Omega' \vdash C}\, \bullet L \qquad q \vdash q}{\Omega, q \twoheadrightarrow (\overleftarrow{F_1^+} \bullet \overleftarrow{F_2^+} \bullet q), q, \Omega' \vdash C}\, \twoheadrightarrow L}\, cut}$$

3.

Case: d^-A.

$$\dfrac{\Omega, q \vdash p \bullet !\overleftarrow{\overrightarrow{A}} \qquad \dfrac{\dfrac{}{\overleftarrow{\overrightarrow{A}}; p \vdash \overrightarrow{\overrightarrow{A}}}\, i.h.}{p \bullet !\overleftarrow{\overrightarrow{A}} \vdash \overrightarrow{\overrightarrow{A}}}\, \bullet L, !L}{\dfrac{\Omega, q \vdash \overrightarrow{\overrightarrow{A}}}{\Omega \vdash q \twoheadrightarrow \overrightarrow{\overrightarrow{A}}}}\, cut$$

Case: $F^+ \multimap F^-$.

$$\dfrac{\dfrac{\Omega, q \vdash \overrightarrow{F^+} \twoheadrightarrow \overleftarrow{F^-} \quad \dfrac{\dfrac{}{F^+ \vdash F^+}\, id \quad \dfrac{}{F^- \vdash F^-}\, id}{\overrightarrow{F^+} \twoheadrightarrow \overleftarrow{F^-}, \overrightarrow{F^+} \vdash \overleftarrow{F^-}} \twoheadrightarrow L}{\Omega, q, \overrightarrow{F^+} \vdash \overleftarrow{F^-}}\, cut}{\dfrac{\Omega, \overleftarrow{F^+}, q \vdash \overleftarrow{F^-}}{\Omega, \overleftarrow{F^+} \vdash \overrightarrow{F^-}}\, i.h.}\, i.h.}{\Omega \vdash \overleftarrow{F^+} \twoheadrightarrow \overrightarrow{F^-}} \twoheadrightarrow R$$

Ass.

■

2007.11.24

I think it would be sensible to have some kind of reversal modality (and corresponding judgment) in ordered logic. Say the two judgments are

$$j ::= \mathsf{b} \mid \mathsf{f}$$

('backwards' and 'forwards') Define $\Omega^\dagger$ by

$$(A_1\ j_1, \ldots, A_n\ j_n)^\dagger = A_n\ j_n^\dagger, \ldots, A_1\ j_1^\dagger$$

where $\mathsf{b}^\dagger = \mathsf{f}$ and $\mathsf{f}^\dagger = \mathsf{b}$. The principal judgment is $\Omega \vdash_{\mathsf{f}} C$, and $\Omega \vdash_{\mathsf{b}} C$ is a derived judgment defined as $\Omega^\dagger \vdash_{\mathsf{f}} C$. The left and right rules for connectives are always defined for hypotheses of the 'forwards' judgment, for a typical connective like $\twoheadrightarrow$ would be like

$$\frac{\Omega, A\,\mathsf{f} \vdash_j B}{\Omega \vdash_j A \twoheadrightarrow B} \qquad \frac{\Psi \vdash_j A \qquad \Omega, B\,\mathsf{f}, \Omega' \vdash_j C}{\Omega, A \twoheadrightarrow B\,\mathsf{f}, \Psi, \Omega' \vdash_j C}$$

and for the reversal operator

$$\frac{\Omega \vdash_{j^\dagger} A}{\Omega \vdash_j \star A} \qquad \frac{\Omega, A\,\mathsf{b} \vdash_j C}{\Omega, \star A\,\mathsf{f} \vdash_j C}$$

The cut principles are:

$$\frac{\Omega \vdash_{\mathsf{f}} A \qquad \Omega_1, A\,\mathsf{b}, \Omega_2 \vdash C}{\Omega_1, \Omega^\dagger, \Omega_2 \vdash C} \qquad \frac{\Omega \vdash_{\mathsf{f}} A \qquad \Omega_1, A\,\mathsf{f}, \Omega_2 \vdash C}{\Omega_1, \Omega, \Omega_2 \vdash C}$$

2007.11.25

The funny thing about focussing in ordered logic is that the asynchronous decomposition looks very different depending on whether you

started on the left or the right. On the left, there is stuff on the left and right fringes of the context, but if you start on the left, all the action is in the middle.

2007.12.8

Trying to push the nicety of the completeness proof back into soundness; not working so well.

2007.12.9

Thought a little about Dan's claim that pattern matching on intensional function spaces is kind of like a positive arrow. Not sure I agree at all. The only way it even starts to work out is if the arrow is asynchronous on both sides, and even then it looks like it wants to turn cointuitionistic on the left. In which case it's something more like $\neg A \wedge B$ rather than $A \Rightarrow B$.

2007.12.11

Frustrating thing about trying to import Luís's notions into HLF-land is the Π_2 heritage of Twelf-like systems.

2007.12.12

Dan and Noam explained the positive arrow stuff to me today. I think I get it, though haven't fully digested it. At the very least I understand better the way that Noam derives asynchronous rules from synchronous ones. Less sure about whether I feel that it's 'modular'.

2007.12.13

For everyone that says "why does everyone have a crush on me" there is someone who says "why does no-one have a crush on me" and vice versa.

On emotions:

(my perhaps over-Westernized notion of) Buddhism teaches some sort of control or abandonment of emotion as attachment. This harmonizes well with an already-present sort of analytic, pragmatic, utillitarian, reductionistic tendency that says: look, if it doesn't do you any good to hold a grudge, and it does do you harm, then *don't*. But the one (religiously enshrined) source of this reaction gets more respect, sometimes, and the other is more often brushed off as geeky roboticness, emotionlessness.

I *have* emotions, but they have harmed me. At greater distance, they harm me less. I have negotiated *this* distance, perhaps inexpertly. (Forgive me.)

It does no good to assert without evidence that it's important to embrace (worse: cling to) or to ignore (better: transcend) emotions. Who knows if it's important or not? It may be that (and it is sometimes asserted that) we can't or shouldn't fight "nature". I could accept the "can't", at least, only if it happens to be true: but it seems that with effort, one can

partially beat it. Without construing it as defeat: one can mold oneself. Here is what bothers me. The smug, pitying belief that if one does not follow certain rules of accepting (as opposed to ignoring or defusing) grief or anger or whatever, then *inevitably* it will come back to bite you. However true this might be, it doesn't *seem* to have *careful* evidence on its side, but rather anecdotes. I could be ignorant of the appropriate evidence.

On the other hand: how could one prove that there is *not* a methodology which allows you to escape this? Here the proponents of emotion-positivism seem to fall back on 'hmph! well, I can't imagine such, and looking for it is clearly wrong-headed'.

Besides which, the line between emotion and non-emotion is always suspect.

Science (or as far as I care the useful part of it) is merely the combination of curiosity and the application of some techniques to avoid obstructions to understanding. It may be identified with these techniques, but perhaps more successful techniques may be discovered.

2007.12.15

The changes on the topic of thesis: lake over water, "exhaustion". Containment leads to intensity. Read as: determine the work that needs to be done, in order that it can be completed with focus.

The function of (a function of) examining divination systems is merely the practice of introducing randomness with a particular flavor, to jostle oneself out of local minima. There is not really 'true' or 'false' randomness, but there are different distributions: it may be useful (or at the very least interesting) to shift my distribution around.

2007.12.22

An essential insight of postmodernism: more truths are more modal (time-indexed, place-indexed, sex-indexed, culture-indexed) than we expected.

An essential insight of "How to Win Friends and Influence People": to be more influential over a person's decisions, *become that person* to a greater extent. When I say to myself 'I shall do this', I believe it because *I* am saying it. When I expressing sincere interest in another person, it blurs the boundary of their identity slightly; I am on their side, I have their interests in mind, etc. To a slightly greater extent, *I am them.*

There is statistical learning on the one hand, and cognitive learning on the other. But there seems to be another form of 'learning' spoken of that is not any more than paraconsistent, but still effective as a means of memorization: call it narrative learning. HtWFaIP is basically constructed out of anecdotes and epigrams, tagged with morals. Nothing prevents us from learning bad lessons this way, but given human psychology, it appears

to be an effective way of (for better or worse) imparting beliefs.

2007.12.29

I can write the following sort of stories. What can be deduced from this?

One.

There was a being who thought himself a god, who thought himself omniscient and omnipotent, because he *knew* the extent of the universe, and every query he posed to himself came back with a certain answer, and each answer was correct, and he *knew* that he knew everything, and if he posed to himself the question of whether he knew everything, the answer came back yes.

But he was wrong. Look: here is our own universe, of which he knew nothing.

Two.

A writer wrote a story, in which a character recited a magic spell that allowed him to escape the story he was in. Immediately, this character appeared next to the writer.

2008.1.1

There are dilemmas of the aggregate. If I do this, it causes this much harm, but that is small. However, if everyone does it, the damage is great. Am I responsible?

The question is: the aggregate may be harming itself. If it is to avoid doing this, it must engage in cognition at a level that corresponds to the problem, which is to say, not at the level of *me*.

Perhaps, however, I am part of this process. I need not know it for it to be true.

2008.1.2

Consider the notion of synthetic judgmental structures.

We specify them by giving a translation of propositional connectives in the synthetic system, into expressions in a known logic. I suppose one would allow some sort of state-machine-ish system to emulate sensitivity to polarity or other (perhaps non-binary) such features.

But we at least need extra expressional functions on the hypothesis and conclusion side, saying at least what *one* hypothesis and *one* conclusion look like, from the outside.

We can then demand without any further use of imagination a limited form of identity and cut.

Say our known logic has a language o of propositions. We ask first of all for propositional functions $H, C : o \to o$ (this $\to$ is really the intensional 'Twelf' arrow). Let p be the new, synthetic language. For each connective

$k : p^n \to p$ we ask for $\overline{k} : o^n \to o$.

So we can easily translate a particular proposition $A : o$ into $\overline{A} : p$ by changing each k in A to $\overline{k}$. Our imagination is temporarily limited to singleton (think: created by monadic injection) contexts, so we translate the sequent

$$A \vdash B$$

to

$$H(\overline{A}) \vdash C(\overline{B})$$

The question is: does the burden of proving this mere transitivity form of cut elimination plausibly (essentially) require the full theorem?

2008.1.3

Claim: If $H(\overline{A}_1) \vdash C(\overline{A}_2)$ and $H(\overline{A}_2) \vdash C(\overline{A}_3)$, then $H(\overline{A}_1) \vdash C(\overline{A}_3)$. If always $C(\overline{A}_2) \vdash H(\overline{A}_2)$ then we're done by appeal to cut-elimination in the original system, but it may not always be this simple.

2008.1.4

SML still suffers from serious annoyances when it comes to implementing long sequences of translations between highly similar languages. What would fix my headaches? Maybe polymorphic variants? I'm not sure.

2008.1.6

$\Gamma \vdash \Delta$ is correct notation when Δ is interpreted conjunctively. If Δ is interpreted disjunctively, this is really $\Gamma; \Delta \vdash \#$.

2008.1.7

The thing that struck me about positive arrow is that, when combined with disjunction, it is simply not familiar logic, by the distributivity over disjunction that as been observed. This means also that Brünnler's deep inference system does not *obviously* extend to disjunction, but maybe there's some tricky way around it that works.

$$\frac{\Gamma, A[] \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma, B[\Delta, A] \vdash C}{\Gamma, A \Rightarrow B[\Delta] \vdash C} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A[\Delta] \vdash A}$$

Cut: If $\Gamma, \Delta[] \vdash A$ and $\Gamma, A[\Delta] \vdash C$, then $\Gamma \vdash C$.

If $\Gamma \vdash \Delta$ and $\Gamma, \Delta[] \vdash C$, then $\Gamma \vdash C$.

Identity: $\Delta[], A[\Delta] \vdash A$.

$\Delta \vdash \Delta$.

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \vee A_2} \qquad \frac{\Gamma, A[\Delta] \vdash C \qquad \Gamma, B[\Delta] \vdash C}{\Gamma, A \vee B[\Delta] \vdash C}$$

alternatively?

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \oplus A_2} \qquad \frac{\Gamma \vdash \Delta \qquad \Gamma, A[] \vdash C \qquad \Gamma, B[] \vdash C}{\Gamma, A \oplus B[\Delta] \vdash C}$$

Identity seems okay for both of these, but I don't expect to be able to prove

$$C \Rightarrow (A \oplus B) \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)$$

but I can prove

$$C \Rightarrow (A \vee B) \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)$$

and

$$C \Rightarrow (A \oplus B) \vdash C \Rightarrow (A \vee B)$$

so cut must fail. Ok, so let's reason this through:

$$\frac{\dfrac{A \oplus B[C], C[] \vdash A \vee B}{C \Rightarrow (A \oplus B) \vdash C \Rightarrow (A \vee B)} \quad \dfrac{A \vee B[C] \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)}{C \Rightarrow (A \vee B) \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)}}{C \Rightarrow (A \oplus B) \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)}\ cut$$

we can call this a principal cut plus one left commutative, and map it to

$$\frac{\dfrac{\star}{A \oplus B[C], C[] \vdash A \vee B} \quad \dfrac{A[C] \vdash C \Rightarrow A \qquad B[C] \vdash C \Rightarrow B}{A \vee B[C] \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)}}{A \oplus B[C] \vdash (C \Rightarrow A) \oplus (C \Rightarrow B)}\ cut$$

where $\star$ is the three premisses $C[] \vdash C, A[] \vdash A \vee B, B[] \vdash A \vee B$.

Hm, suspect. Let's try proving the commutative case for $\oplus L$ in isolation. We have

$$\frac{\dfrac{\Gamma, \Psi \vdash \Delta \qquad \Gamma, \Psi, X_1[] \vdash A \qquad \Gamma, \Psi, X_2[] \vdash A}{\Gamma, \Psi, X_1 \oplus X_2[\Delta] \vdash A} \qquad \Gamma, A[\Psi] \vdash C}{\Gamma, X_1 \oplus X_2[\Delta] \vdash C}$$

we get $\Gamma, X_i[] \vdash C$ for both i by cut, but then we try to do $\oplus L$ again and we get only

$$\frac{\Gamma, \Psi \vdash \Delta \qquad \Gamma, X_1[] \vdash C \qquad \Gamma, X_2[] \vdash C}{\Gamma, \Psi, X_1 \oplus X_2[\Delta] \vdash C}$$

Similarly I would expect a Dyckhoff-like implementation of the ordinary arrow

$$\frac{\Gamma, A[] \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma, B[] \vdash C}{\Gamma, A \to B[\Delta] \vdash C}$$

to fail in roughly the same way.

Suppose I did this with linear logic, so that & is surely negative. Can I create a synthetic connective out of & and 'positive $\multimap$'?

$$\frac{\Gamma, A[] \vdash B \qquad \Gamma \vdash C}{\Gamma \vdash (A \multimap B) \,\&\, C} \quad \frac{\Gamma, C[\Delta] \vdash D}{\Gamma, (A \multimap B) \,\&\, C[\Delta] \vdash D} \quad \frac{\Gamma, B[\Delta, A] \vdash D}{\Gamma, (A \multimap B) \,\&\, C[\Delta] \vdash D}$$

$$\frac{\Gamma, A[] \vdash B \qquad \Gamma, A[] \vdash C}{\Gamma \vdash A \multimap (B \,\&\, C)} \quad \frac{\Gamma, C[\Delta, A] \vdash D}{\Gamma, A \multimap (B \,\&\, C)[\Delta] \vdash D} \quad \frac{\Gamma, B[\Delta, A] \vdash D}{\Gamma, A \multimap (B \,\&\, C)[\Delta] \vdash D}$$

But I do seem to have some evidence that even now, positive $\multimap$ is not negative in its first argument:

$$\frac{\Gamma, A[] \vdash C}{\Gamma \vdash (A \,\&\, B) \multimap C} \quad \frac{\Gamma, B[] \vdash C}{\Gamma \vdash (A \,\&\, B) \multimap C} \quad \frac{\Gamma, C[\Delta, A \,\&\, B??] \vdash D}{\Gamma, (A \,\&\, B) \multimap C[\Delta] \vdash D}$$

for it's not clear how to write the left rule. Note that we get the same right rules from synthesizing $(A \multimap C) \oplus (B \multimap C)$ but this is not bientailed by $(A \,\&\, B) \multimap C$.

Is it positive in its first argument?

$$\frac{\Gamma, A[] \vdash C \quad \Gamma, B[] \vdash C}{\Gamma \vdash (A \oplus B) \multimap C} \quad \frac{\Gamma, C[\Delta, A] \vdash D}{\Gamma, (A \oplus B) \multimap C[\Delta] \vdash D} \quad \frac{\Gamma, C[\Delta, B] \vdash D}{\Gamma, (A \oplus B) \multimap C[\Delta] \vdash D}$$

These seem like they might be okay, but they are the same rules as I'd get from synthesizing $(A \multimap C) \,\&\, (B \multimap C)$ — oh, which **is** bientailed by $(A \oplus B) \multimap C$!

Okay, so is it really positive on the outside?

$$\frac{\Gamma, A[] \vdash B}{\Gamma \vdash (A \multimap B) \oplus C} \quad \frac{\Gamma \vdash C}{\Gamma \vdash (A \multimap B) \oplus C} \quad \frac{\Gamma, C[\Delta] \vdash D \qquad \Gamma, B[\Delta, A] \vdash D}{\Gamma, (A \multimap B) \oplus C[\Delta] \vdash D}$$

Seems disturbingly like it. In conclusion, I find myself able to believe in two arrows in this system, one $(+, -) : -$, and one $(+, +) : +$, but **both** of them distribute strangely over disjunction.

2008.1.9

Dan and Noam sidestep the above question by making the syntactic type of things that are arguments to positive arrows totally separate, and therefore not clearly positive or negative.

2008.1.10

The Πw-at-kind-level thing still seems like a plausible intuition for how to think about coverage checking etc.

Does *SOME* in block declarations demand the expression is closed? Probably not.

2008.1.11

The difference between free and existential variables is subtle. Should think more about whether unification can stand up to circularity as long as simple types are maintained.

2008.1.12

Couldn't I treat all the free variables at once as a big multi-pi, and then just ask the question of whether they're topological sorted after the fact?

2008.1.13

The free and existential variables are certainly "intertwined" in the sense that their types may involve the other.

Thus:

We begin with a constant declaration. We look at its classifier. We infer a simple type for it. This might as well go so far as what Frank and Kevin actually do habitually call a *simple* type, not what I've been lazily calling that — as opposed to a mere skeleton. That is, actually figure out at least the family for everything, but not the indices. If simple type inference cannot even figure out the family, we'll be left with type-level evars at the very end anyhow, which won't do us any good.

So in some sense FVs and EVs are both modal. For all underscores, we make up an evar of functional type with included dependencies on all local bound variables. For FVs, their type includes evars for each index.

What happens if you explicitly indicate a dependency on a bound variable? Should the system duplicate it and yield a non-pattern?

Experimentally: this is what twelf does.

```
o : type.
a : o -> type.
c : {y : o} {x : o} a (_ y).

==>
```

```
[Opening file /tmp/a.elf]
o : type.
a : o -> type.
c : {X1:o -> o -> o -> o} {y:o} {x:o} a (X1 y x y).
[Closing file /tmp/a.elf]
val it = OK : Twelf.Status
-
```

So I seem to be left with a context of variables about which I am curious as to whether a well-typed instantiation of a certain subset of them leads to certain equations holding.

After finding the most general such instantiation (if it exists) I can then ask whether there is a topological sort of the resulting context.

This context 'knows' the full types of all its variables, but they may of course be expressions that involve instantiatable variables from precisely that context.

Say that every variable intrinsically, syntactically knows its simple type.

The context of metavariables looks like

$$\Delta ::= (u_1, \ldots, u_n) : (A_1, \ldots A_n)$$

Damn — there seems to be three dispositions of variables that I might care about for the definition of solutions, then; variables that are never instantiated (FVs) and apart from that, EVs that are vs. are not instantiated right now.

2008.1.14

We suppose there to be existential variables and free variables at various contextual types. Every appearance of these guys in the expression occurs under a substitution. The only real difference between a substitution and a spine is the associativity of them; and that substitutions sort of carry a functional interpretation so that inverting them makes sense.

We might as well do everything with substitutions, right?

Anyway, the unification query names a subset of the evars, and asks: *what* are the well-typed instantiations of them that extend to well-typed instantiations of all evars (not necessarily closed) so that some resulting equations hold?

Equations can be decided syntactically.

The substitution principle is something like: if $\Delta, \Gamma \vdash J$, and $\Delta, \Psi \vdash \sigma : \Gamma\sigma$, then $\Delta, \Psi \vdash J\sigma$.

The type invariant is: every equation is well-typed, modulo all the equations being solved.

We start this invariant off by having equations that are well-typed *period*, modulo nothing, right?

2008.1.15

Uncertainty — funny that the natural word, and all the synonyms I can think of it, are so negatively defined. Not knowing for sure. Imperfect knowledge. Unclear, undetermined. etc.

The things we negate here are the exception! Uncertainty is the rule; the blankness of the future, of distant places and times, the not-clarity of the not-here, not-now.

The thesaurus provides me some good positive words for this phenomenon: opacity, vagueness, ambivalence, ambiguity, conjecturality. The notions I can get at here are positive notions of the *un-seeability* of things, but also their *multi-valuedness*, conveying the feel of all the endlessly ramifying possible worlds sitting in the same space, and also our (mere) ability to guess at these possibilities.

2008.1.16

Saw Derek's talk about his and Andreas Rossberg's mixin module calculus. There's something pleasing about the initial simplicity of it, but it soon got quite complicated. Sadly the calculus presented int he talk was the EL, not the IL. The latter still involves the mysterious sort-of-stateful backpatching that I have never understood.

One argument for why the EL *cannot* be so easily turned into a reasonable IL is that there is no evident type system for it, it being all at one universe, so to speak. Nonetheless I think one might hope for a system of stratification that leaves it (parametrically) looking just about as simple, much like the LF stratification into types and kinds and so forth.

2008.1.17

I still think module systems as we conceive them are "too focused", and lump too many things together. I wonder what makes sense in the domain of mere name-management. The namespace lifting operator $\{\ell = mod\}$ from yesterday, for instance, is interesting.

2008.1.19

The only difference between fvars and evars seems to be that the fvars are not subject to further substitution. So closedness of a substitution is merely a convenience that says it's the last substitution that will take place right now — even though, of course, fvars will get abstracted to Πs, which will 'allow' substitution later via application.

It is still necessary to be careful what is meant by a fully well-typed solution to some equations. If I introduce extra fvars, they are *ab initio*

merely simply typed. So I say:

A *solution* to a unification problem $\Psi; \Delta \vdash P$ consists of a (tacitly simply-well-typed) substitution θ for Δ that leaves all of P true. The codomain of θ may mention fvars not in Ψ. This substitution θ is a *well-typed solution* of P iff there *exists* an extension of Ψ to Ψ' such that $\Psi'; \Delta \vdash \theta : \Delta$.

2008.1.20

Inversion:

$$i ::= r \mid f$$

(rigid or flex)

$$n\{\sigma\}^i_u = \begin{cases} m & \text{if } m\{\sigma\} = n; \\ \_ & \text{if no such } m. \end{cases}$$

$$v[\sigma]\{\rho\}^i_u = v[\sigma\{\rho\}^i_u]$$

$$u[\sigma]\{\rho\}^r_u = \_$$

$$u[\sigma]\{\rho\}^f_u = \mathbf{abort}$$

$$(x \cdot S)\{\sigma\}^{-1}_u = x \cdot (S\{\sigma\}^f_u)$$

$$(c \cdot S)\{\sigma\}^{-1}_u = c \cdot (S\{\sigma\}^r_u)$$

(but treat non-local variables like constants) _ is like an exception that bubbles up to substitution elements.

2008.1.21

Thinking about a statistical puzzle ("the paradox of early stopping"?) Gustavo told me. Remarkable how shaken the validity of scientific results can be if we don't know all of the experiments that 'fail'.

2008.1.22

I was incorrectly imagining that there was a three-way difference between variables, *local* variables, and constants in unification. This is silly. Unifiers are relatively closed; there are only variables (which might well be called local) and constants.

For non-pattern substitutions σ, even when ξ is a pattern, it is not safe to reject on occurs-check in situations like

$$u[\xi] \doteq x \cdot (\cdots u[\sigma] \cdots)$$

because as a counterexample you might get (for $u :: f : o \to o \vdash o$)

$$u[x/f] \doteq x \cdot (u[\lambda y.k/f])$$

which allows $u \leftarrow f.f \cdot k$ as an instance.

I suppose it's still ok to reject

$$u[\xi] \doteq x \cdot (\cdots u[\xi'] \cdots)$$

though, as long as $u[\xi']$ occurs rigidly?

2008.1.23

Does it make any sense to attach scalar multiples to lambda terms to affect merely how β-reductions are counted so as to effect a sort of strong diamond property?

2008.1.24

The unification algorithm consists of these kinds of steps:

1. Homomorphic decomposition of lambda, spine application and cons.
2. Inversion (rigid and weakly-flex)
3. Projection (actually projecting out _ arguments of evars that occur rigidly)
4. Intersection ($u[\xi] \doteq u[\xi'] \mapsto u[\xi \cap \xi'] \doteq u[\xi \cap \xi']$)
5. Extra Occurs-check ($u[\xi] \doteq x \cdot (\cdots u[\xi'] \cdots) \mapsto \bot$)

Type preservation and solution preservation are easy for 1, 4, 5. Solution preservation seems easy for 3, but what about types? What happens when the projected-out variable occurs in the type later on?

$$v :: (o, a\ 1 \vdash o)$$

$$x : o, y : a\ x \vdash u[y] = c \cdot v[y, x]$$

Hm, it seems that this can't happen for the pure pattern fragment, but it can certainly happen otherwise.

$$u :: (a\ u[\mathsf{id}] \to a\ u[\mathsf{id}] \vdash o)$$

$$v :: (o, a\ 1 \to a\ 1 \vdash o)$$

$$y : a\ u[\mathsf{id}] \to a\ u[\mathsf{id}] \vdash u[y] \doteq c \cdot v[y, u[\mathsf{id}]]$$

$$\mapsto\vdash u \leftarrow c \cdot v[1, \_]$$

Here we can't really project out the underscore in v since the type of y still depends on it. Maybe we simply refrain from making such a projection if it doesn't work out? I don't think there would be an extra computation to see if it was possible. I think it would just pop out as the attempt to

compute what the type of the new evar is resulting in an exception being thrown.

2008.1.25

Ok, so say inversion only deals with bound variable replacement and doesn't do anything else. Like:

$$u[x, y] \doteq c \cdot (x; v[y, u[x, z]]) \mapsto u \doteq c \cdot (1; v[2, u[1, \_]])$$

and only secondarily to we do occurs-check and stuff — in fact, we postpone carrying out the known instantiation until we know that there are no occurrences of u on the right.

Underscore cannot always bubble up in nonpatterns past 'locally' bound variables in a different sense of locally. Consider

$$u[] \doteq c \cdot v[\lambda z.k, \lambda y.y \cdot u[]]$$

we do have a solution in $v \leftarrow 2 \cdot 1$, so we should make a move to

$$u \leftarrow c \cdot v[\lambda z.k, \lambda y.y \cdot \_]$$

not

$$u \leftarrow c \cdot v[\lambda z.k, \_]$$

Projection takes place *when* the types can be so projected, but since Πs are linearized, we can eagerly attempt to project rightmost things in a context, which might make more leftward things projectible later.

Really _ is a root, I think; I ought to lambda abstract it when it is standing for a bunch of lambdas, but whatever.

So my new set of things is

1. Homomorphic decomposition of lambda, spine application and cons.
2. Inversion
3. Projection (actually projecting out _ arguments of evars that occur rigidly)
4. Intersection Projection (project out n from $u \doteq u[\xi, n, \xi']$ when $n \neq |\xi|$)
5. Pattern occurs-check ($u[\xi] \doteq x \cdot (\cdots u[\xi'] \cdots) \mapsto \bot$ if u occurs rigidly on the right)
6. Rigid head occurs-check ($u[\xi] \doteq c \cdot (\cdots u[\sigma] \cdots) \mapsto u[\xi] \doteq c \cdot (\cdots \_ \cdots)$)
7. Pushing up underscores.

8. Changing $u \doteq R$ to $u \leftarrow R$ and carrying out modal substitution $\{R/u\}$.

Suppose $u :: A_1, \ldots, A_n \vdash A$. We hear from somewhere else that u's instantiation cannot possibly use its ith argument. So we set up a new variable v with type

$$v :: A_1, \ldots, A_{i-1}, A_{i+1}\theta, A_{i+2}\uparrow\theta, \ldots \vdash A\uparrow^{n-i}\theta$$

where $\theta = \{\_.\mathsf{id}\}$, and $\uparrow\sigma = 1.\uparrow \circ \sigma$, and we add an equation

$$u \leftarrow v[1, \ldots, \hat{i}, \ldots, n]$$

Now: could this substitution go through even when a variable *is* used deeply under some other evar, resulting in a captured exception?

I can check whether a given substitution is well-typed, before I check that the classifiers *are* types. So I can say things like "for all well-typed substitutions, the following thing is well-typed", also "for all well-typed substitutions that satisfy P, the following thing is well-typed".

2008.1.27

Recall that economic value is torsorial — how can one sensibly speak of derivatives with bounded exposure about goods that may themselves be perishable? The value of fiat currency derives from peculiarly human recursive beliefs, just like art.

2008.1.28

Does game semantics merely dress up quantifier alternation in cute clothing? I think reading Colin Stirling's higher-order matching work has a good chance of disabusing me of precisely this cynicism.

Trying to figuring out what the typing invariant on underscored expressions should be is still vexing. I cannot assume that they only appear to the right of $u \leftarrow M$ if I want to actually carry out substitutions of such — which is the whole point.

2008.1.29

Figuring out the typing invariant for unification with underscore is infuriating.

2008.1.30

Harry Mairson gave a talk on how kCFA is EXPTIME-complete. I didn't quite see how the $k \geq 0$ came into play. For $k = 0$ it happens to be PTIME-complete.

2008.1.31

Consider type theories that capture polynomial-time computation; can they be refined to say anything about particular-degree polynomials?

Mairson's assertion that all static analyses are essentially abstract interpretation sticks in my mind. Don't know whether to agree with it or not.

* * * * * *

Try the following invariant for typing in unification:

Given $M_1 \doteq M_2 \in P$. It is well-formed if there is a Γ and A such that for any θ_1 that is a *well-typed* instantiation of every free evar that results in no underscores in M_1 and M_2, we have $\Gamma \vdash_P M_i\theta_1 \Leftarrow A$. The $B \equiv_P B'$ deep inside that judgment essentially: means for any θ_2 that is a (*not* necessarily well-typed) instantiation of all the evars that satisfies the equations in P, then $B\theta_2 = B'\theta_2$.

2008.2.1

Rename the θ_1, θ_2 above to θ_t, θ_e for *t*yped and *e*quational substitutions. For the spine cons case, we have by assumption

$$\frac{\Gamma \vdash_P M_i\theta_t \Leftarrow A \qquad \Gamma \vdash_P S\theta_t : [M_i\theta_t/x]B > C_i}{\Gamma \vdash_P (M_i; S_i)\theta_t : \Pi x{:}A.B > C_i}$$

for some $C_1 \equiv_P C_2$. But we should have $M_1 \equiv_P M_2$, so $M_1\theta_t \equiv_P M_2\theta_t$?

This doesn't really work — I've got the quantification mixed up. I am introducing a $\forall$ in the two premises, and I let two θ_t be given, and then I don't know how to reconcile them in the $\forall$ elimination for the conclusion.

Here's another attempt. An entire set of underscore-mentioning equations P is well-formed if: for any *possibly open* substitution θ_t that is well-typed, not necessarily satisfying of all the equations, but which eliminates all underscores, $P\theta_t$ is well-formed in the sense of $\vdash_{P\theta_t}$.

Then, we start by considering $(M_1, S_1) \doteq (M_2, S_2) \mapsto M_1 \doteq M_2 \wedge S_1 \doteq S_2$. We want to show the latter is well-formed, so let a θ_t be given. Since the types of evars have not changed, this is also a valid typed substitution for the former state. By inversion we see

$$\frac{\Gamma \vdash_{P\theta_t} M_i\theta_t \Leftarrow A \qquad \Gamma \vdash_{P\theta_t} S\theta_t : [M_i\theta_t/x]B > C_i}{\Gamma \vdash_{P\theta_t} (M_i; S_i)\theta_t : \Pi x{:}A.B > C_i}$$

(For some $C_1 \equiv_{P\theta_t} C_2$) But $M_1\theta_t \equiv_{P\theta_t} M_2\theta_t$, so we can transfer the spine typing to the appropriate type.

2008.2.2

The thing from yesterday seems to be working well so far.

2008.2.3

Hit a snag: instantiation fucks up types in the context. Still searching for a workaround.

2008.2.4

Here's an idea. Define a notion of 'simple development' of a unification problem that allows introduction of new variables, new underscore-free well-typed assignments, and replacement of equals with equals. The invariant is: every underscore-free simple development of the current unification problem is well-typed.

2008.2.5

Here's a gadget that seems sort of like a free strict ω-category.

Call it a 'dicomplex' by analogy with ditopologies.

The data of one consists of a set C_n of n-cells for each n, and again for each n there are maps $\mathrm{dom}, \mathrm{cod} : C_{n+1} \to P_n$. The set P_n is defined recursively from the C_n. First of all we have $P_0 = C_0$, and P_{n+1} is going to be the set of pasting diagrams that arise from syntactic expressions over C_{n+1}.

What are these syntactic expressions? A candidate π for P_n is made from

$$\pi ::= c^n \mid \pi \circ_m \pi \mid \mathsf{id}_p$$

where p is an actual path from P_{n-1}. When is one of these well-formed?

$$\frac{\mathrm{dom}\, c^n = p_1 \qquad \mathrm{cod}\, c^n = p_2}{\vdash c^n : p_1 \Rightarrow_n p_2}$$

$$\frac{}{\vdash \mathsf{id}_p : p \Rightarrow_n p}$$

$$\frac{\vdash \pi_1 : p_1 \Rightarrow_n p_2 \qquad \vdash \pi_2 : p_2 \Rightarrow_n p_3}{\vdash \pi_2 \circ_0 \pi_1 : p_1 \Rightarrow_n p_3}$$

$$\frac{\vdash \pi^i : p_1^i \Rightarrow_n p_2^i \qquad \vdash p_i^2 \circ_m p_i^1 : \tau^{n-1} \qquad (\forall i \in \{1,2\})}{\vdash \pi^2 \circ_{m+1} \pi^1 : (p_1^2 \circ_m p_1^1) \Rightarrow_n (p_2^2 \circ_m p_2^1)}$$

A τ^n is something of the form $p \Rightarrow_n p'$ for $p, p' \in P_{n-1}$.

Now we want to interpret such a candidate as a graph of sorts. Notice that P_n also supports dom and cod operations.

* * * * * *

Erg, let me start over.

To describe a dicomplex, we provide sets C_n and maps $\mathrm{dom}, \mathrm{cod}\, C_{n+1} \to P_n$, which are to satisfy certain axioms. First, define the P_n. An element of P_n is called an n-diagram.

A 0-diagram is a set X, and a map $\ell : X \to C_0$. An $n+1$-diagram is a tuple (X, ℓ, p, d, c) where

- X is a set
- ℓ is a map $X \to C_{n+1}$
- $p \in P_n$
- d, c are both functions such that $d(x)$ and $c(x)$ are both morphisms from $\mathrm{dom}(\ell\ x)$ to p, for any $x \in X$

A 0-diagram morphism (X, ℓ) to (X', ℓ') is a function $f : X \to X'$ such that $\ell' \circ f = \ell$. An $n+1$-diagram morphism (X, ℓ, p, d, c) to (X', ℓ', p', d', c') is a pair consisting of $f : X \to X'$ and $g : p \to p'$ such that

- $\ell' \circ f = \ell$
- $d' \circ f = \lambda x.g \circ d(x)$
- $c' \circ f = \lambda x.g \circ c(x)$

The intuition behind the axioms is that I want to require the domains and codomains in a dicomplex to be 'connected' pasting diagrams, which are generated from composition, identities, and cells. Connected pasting diagrams have a clear notion of domain and codomain themselves, so that I can require that the domain and codomain *of* the domain and codomain are the same.

2008.2.6

Let me try to construct a counterexample to the type soundness of unification as I understand it.

It would have to involve

$$P \wedge u \leftarrow R \mapsto [R/u]^1 P \wedge u \leftarrow R$$

I suppose beforehand that there for any well-development of the former that results in no underscores, the result is well-typed. I wish to show this about the latter. I can take any development of the latter and mostly imitate it in the former: the only thing I cannot to is imitate the substitution of R for u itself, since I don't generally know that R is well-typed. If the development apart from R/u already eliminates all underscores, then I do. So I must worry that it doesn't.

Now u could occur in another equation somewhere, or else in a type in Δ. This being unless I don't need to push developments through to Δ, not sure about that. I will have to deal with the case of other equations at the very least.

So my situation looks like:

$$P \wedge Q(u[\sigma]) \wedge u \leftarrow R \mapsto P \wedge Q(R[\sigma]) \wedge u \leftarrow R$$

Hmm... Here's an idea: a development step is licensed if the terms are well-typed modulo the equations *after* adding it.

2008.2.9

Summary of approaches tried so far:

1. Naive attemps to find typing invariant on underscore expressions
2. Early quantification that failed to separate equality and typing
3. Separate quantificational approaches
   (a) For all underscore-eliminating substitutions (one for each equation; fails at cons case)
   (b) For all underscore-eliminating substitutions (global; fails at substitution case)
   (c) For all developments (various definitions of 'development'; also fails at substitution)
   (d) For all generalized projections (couldn't figure out good definition)
   (e) For all entailed equalities (fails at the outset)
4. Attempt to rewrite every underscore-using unification trace to one that doesn't.
   (a) By analyzing and eliminating circularity (works for purely rigid problems)
      i. Without extra nondeterministic guessing of, e.g. generalized projections
      ii. With them
   (b) **NEW**: By blatantly deunderscoring failed occurs-checks and substituting 'forbidden fvars' for underscores arising from inversion.

2008.2.10

I tried to get rid of the 'forbidding' itself, but it seems to be easier to keep it — that way I can push the identical equational theory down through both the *actual* trace and the one that is simulated in parallel so as to show preservation of types. While creating them during inversion, one may need to create them at higher type and apply them to local variables that *were* preserved in order to get the typing to work right.

For underscores arising from occurs-checks I just carry out one substitution and leave it at that. This move would be no good in the actual

algorithm of course because of termination concerns, but the 'simulated' trace simply tracks the original, and by construction ends the same way.

2008.2.13

The totality of patterns is useful after all. Consider the contexts $\Gamma = x : o, y : a\ x, z : a\ x \to o$ and $\Psi = w : o, w' : a\ w, v : o, v' : a\ v \to o$.

Faced with the equation

$$u[z.\_.y.\_] \doteq z\ y$$

(which is well-typed because we can fill in both underscores with x) we would like to say $u \leftarrow v'\ w'$, but this is not well-typed.

2008.2.14

Dan Licata talking again about his and Noam's work. Definitional Reflection a la Schröder-Heister feels very much like merely offering a *graph-theoretic* way of defining propositions.

2008.2.15

A return to thinking about multidimensional graphs.

We define three notions indexed by natural numbers: graphs, paths, and path morphisms. An n-path exists in an n-graph. An n-path morphism is between two n-paths. The homset of morphisms between n-paths P and P' is written $n\mathrm{Path}(P, P')$.

- A 0-graph is a set C.
- A 0-path in a 0-graph C is a set X and a map $\ell : X \to C$.
- A 0-path morphism $(\ell : X \to C) \to (\ell' : X' \to C)$ is a morphism $f : X \to X'$ such that $\ell(x) = \ell'(f(x))$ for all x.
- An $(n+1)$-graph is a tuple $(G, C, \mathrm{dom}, \mathrm{cod})$ where G is an n-graph, C is a set, and dom, cod are maps that take elements of C to n-paths in G.
- An $(n+1)$-path in an $(n+1)$-graph of the form $(G, C, \mathrm{dom}, \mathrm{cod})$ is a tuple (P, X, ℓ, d, c) where
  - P is an n-path in G
  - X is a set
  - ℓ is a map $X \to C$
  - d is a function of type $\Pi x{:}X.n\mathrm{Path}(\mathrm{dom}(\ell(x)), P)$
  - c is a function of type $\Pi x{:}X.n\mathrm{Path}(\mathrm{cod}(\ell(x)), P)$

- An $(n+1)$-path morphism $(P, X, \ell, d, c) \to (P', X', \ell', d', c')$ is a tuple (f, g) where
  - $f : X \to X'$
  - $g : n\mathrm{Path}(P, P')$
  - $\ell(x) = \ell'(f(x))$
  - $d'(f(x)) = g \circ d(x)$
  - $c'(f(x)) = g \circ c(x)$

(Categorifying, I might think d, c were sort of natural transformations.)

Let's try to define connected paths.

One judgment is $f : A \to B$ ('f is a connected path from A to B') where f is an $(n+1)$-path, and A, B are n-paths. We will arrange in this case for there to be a homomorphism of A and B into the underlying n-path of f.

The other one is $P : o$, ('P is a connected 0-path') which is true just in case the underlying set of P is a singleton. τ stands for either $A \to B$ or o.

$$\frac{P : \tau}{\mathsf{id}_P : P \to P}$$

There are no $(n+1)$ cells at all of id_P. The homomorphisms from P and P are the identity ones.

$$\frac{f : A \to B \qquad g : B \to C}{g \circ_0 f : A \to C}$$

We have an $f = (P, X, \ell, d, c)$ and $g = (P', X', \ell', d', c')$ and maps $i_A : n\mathrm{Path}(A, P), i_B : n\mathrm{Path}(B, P)$ and $j_B : n\mathrm{Path}(B, P'), j_C : n\mathrm{Path}(C, P')$. I think what we do then is take the two evident arrows $n\mathrm{Path}(B, P + P')$ that we can hack out of i_B, j_B and coproduct injections, and take their coequalizer $n\mathrm{Path}(P + P', Q)$. We can string along the old c, c', d', d' by tacking on injections and sending them to Q via the coequalizer. and same thing with i_A and j_C. The label set for the result is just $X + X'$, and we similarly coproduct together $\ell + \ell'$ etc.

$$B_1 \circ_n A_1 : C \to D \qquad B_2 \circ_n A_2 : C \to D$$
$$\frac{g : B_1 \to B_2 \qquad f : A_1 \to A_2}{g \circ_{n+1} f : (B_1 \circ_n A_1) \to (B_2 \circ_n A_2)}$$

We have an $f = (P, X, \ell, d, c)$ and $g = (P', X', \ell', d', c')$ and maps

$$i_1 : n\mathrm{Path}(A_1, P), i_2 : n\mathrm{Path}(A_2, P)$$

$$j_1 : n\mathrm{Path}(B_1, P'), j_2 : n\mathrm{Path}(B_2, P')$$

Then I guess I need to take a big colimit. I should also be inductively guaranteeing that there are embeddings of arrows into their composition so this is possible.

2008.2.16

Yeah, the colimit construction seems right — I want to lift (via *id*) things of lower dimension.

The funny thing about the Baez-Dolan 'periodic table' in this setting in that if you look at a twice-monoidal set (i.e. a commutative monoid) the commutativity arised from the fact that you can't anchor down the two-cells to any particular one-cell — the only one that there is is the identity on the unique object, so you can't tell the different paths apart.

2008.2.18

There's a notion of 'admissibility' mentioned in Neelk's thesis proposal that seems interesting – it is supposed to have come from Lars Birkedal and some coauthor. Sadly it's not as symmetric as I thought, passing from arbitrary functions to extensions to continuous functions: all the approximation seems to happen in the second step.

2008.2.20

Two uses for goals: as information for error-correction during communication, and as predictions for the sake of self-directed 'machine-learning'.

2008.2.21

Went to a talk by some guy about mechanisms and cognition. Pretty low-content. I am frustrated at notions like 'embodied' vs. 'nonembodied' computation because they seem contentless — or rather, that their content is mushy and subjective. Like, plainly, a machine with eyes and arms and legs interacts with the world in a very complicated way, but it's not as if a laptop with a monitor and keyboard *doesn't* interact with the world. It's just that it can't move about very effectively.

2008.2.22

I feel like I should be able to identify the 'twist' arrow in the ω-graph analogue of a braided monoidal category. It would be something with 2- and 3-cells, and trivialized at 0- and 1-cells. The only 1-cell is the identity arrow $\bullet \to \bullet$. All 2-cells are therefore horizontally and vertically composable. Suppose there are two basic 2-cells x and y... actually, I can't find any evidence that a twist exists, because as things are set up, $x \circ y$ and $y \circ x$ are just plain identical. Maybe this has to do with the fact that every bicategory is equivalent to a 2-category, but not every tricategory is equivalent to a 3-category?

Nonetheless by using 1-categorical concepts (of taking colimits of paths to define composition) I might have got an adequate notion of 2-graphs. Maybe this can be pushed up somehow?

2008.2.23

What is the cut elimination theorem for ordered logic like in HLF?

If $\Omega \vdash A$ and $\Omega_L, A, \Omega_R \vdash C$, then $\Omega_L, \Omega, \Omega_R \vdash C$.

```
ca : {a} {b} {c}
     (conc A @ a)
  -> ({d} hyp A @ d -> conc C @ (b * d * c))
  -> (conc C @ b * a * c)
  -> type.
```

2008.2.24

'Open-ended' equality for LF.

Syntax:

$$\begin{array}{rrl} \text{Expressions} & E & ::= M \mid \tilde{A}(E) \\ \text{Types} & A & ::= \cdots \mid \{A\} \\ \text{Terms} & M & ::= \cdots \mid \{E\} \end{array}$$

Judgments:

$$\Gamma \vdash \tilde{X} : X_1 \sim X_2$$

$$\Gamma \vdash E \div A$$

Rules:

$$\frac{\Gamma \vdash E \div A}{\Gamma \vdash \{E\} \Leftarrow \{A\}} \qquad \frac{\Gamma \vdash M \Leftarrow A}{\Gamma \vdash M \div A} \qquad \frac{\Gamma \vdash E \div A_1 \qquad \Gamma \vdash \tilde{A} : A_1 \sim A_2}{\Gamma \vdash \tilde{A}(E) \div A_2}$$

$$\frac{\Gamma, y : Y_1 \sim Y_2 \vdash \tilde{X} : X_1 \sim X_2 \qquad \Gamma \vdash E \div Eq(Y_1, Y_2)}{\Gamma \vdash \mathbf{let}\, y = E \,\mathbf{in}\, \tilde{X} : X_1 \sim X_2}$$

2008.2.25

A funny thing about base-type polymorphism is that the obvious definition of Leibniz equality isn't symmetric — going down an order in the types I can get a reflect, but it's not obvious whether this process has a fixpoint.

2008.2.26

Turns out the higher-order logic programming thing I thought of is totally old hat to lambda prolog hackers.

2008.2.27

Think I have an easier way of thinking about the Stirling algorithm. Dunno whether it will stand up to his later development, but it seems largely isomorphic for the special case of there being only first-order constants.

Set up the usual contextual business by saying

$$\begin{array}{rrl} \text{Normal Terms} & M & ::= \lambda\hat{\Psi}.R \\ \text{Atomic Terms} & R & ::= x \cdot S \mid f(R,R) \mid c \\ \text{Substitutions} & \sigma & ::= \mathsf{id} \mid M.\sigma \end{array}$$

and then add

$$\begin{array}{rrl} \text{Preatomic terms} & P & ::= (M,\sigma) \mid R \\ \text{Lookup Table} & \beta^\pi & ::= \cdot \mid \beta^\pi[(\beta^{\bar\pi}, M)/x] \\ \text{Lookup Table Pair} & \gamma & ::= (\beta^0, \beta^1) \\ \text{Polarity} & \pi & ::= 0 \mid 1 \end{array}$$

We write projection from pairs as γ_π, and polarity flipping as $\bar\pi$. A closed atomic term is written r. A state is a tuple (P, r, γ, π). The interpretation of a state is the proposition $[\gamma;\pi]P = r$. The left side is defined by

$$[\gamma;\pi](M,\sigma) = [\gamma_{\bar\pi}M \mid \gamma_\pi\sigma]$$

$$[\gamma;\pi]R = \gamma_\pi R$$

We define winning states inductively.

$$\frac{\vdash (R, r, \gamma[(\gamma_{\bar\pi},\sigma)/\hat\Psi], \pi)\,\mathsf{win}}{\vdash ((\lambda\hat\Psi.R,\sigma), r, \gamma, \pi)\,\mathsf{win}} \qquad \frac{}{\vdash (c,c,\gamma,\pi)\,\mathsf{win}}$$

$$\frac{\vdash (R_1, r_1, \gamma, \pi)\,\mathsf{win} \qquad \vdash (R_2, r_2, \gamma, \pi)\,\mathsf{win}}{\vdash (f(R_1,R_2), f(r_1,r_2), \gamma, \pi)\,\mathsf{win}}$$

$$\frac{(\beta^{\bar{\pi}}, M)/x \in \gamma_\pi \qquad \vdash ((M,\sigma), r, \gamma \leftarrow \beta^{\bar{\pi}}, \bar{\pi})\,\mathsf{win}}{\vdash (x[\sigma], r, \gamma, \pi)\,\mathsf{win}}$$

Where $\gamma[(\gamma_{\bar{\pi}}, \sigma)/\hat{\Psi}]$ and $\gamma \leftarrow \beta^{\bar{\pi}}$ are hopefully obvious abbreviations for substitution operations.

2008.2.29

Even better: A state is a tuple (R, r, γ). The interpretation of a state is the proposition $\gamma R = r$. The notion of winning state

$$\frac{}{\vdash (c, c, \gamma)\,\mathsf{win}} \qquad \frac{\vdash (R_1, r_1, \gamma)\,\mathsf{win} \qquad \vdash (R_2, r_2, \gamma)\,\mathsf{win}}{\vdash (f(R_1, R_2), f(r_1, r_2), \gamma)\,\mathsf{win}}$$

$$\frac{(\gamma', \lambda\hat{\Psi}.R)/x \in \gamma \qquad \vdash (R, r, \gamma' + ((\gamma,\sigma)/\hat{\Psi})\,\mathsf{win}}{\vdash (x[\sigma], r, \gamma)\,\mathsf{win}}$$

2008.3.3

Can't trust the constant rule that I have necessarily. Counterexample:

$$\lambda f.u[f] \doteq c\ (f\ (u[\lambda x.k]))$$

has solution $u \leftarrow c\ (f\ (ck))$.

2008.3.4

I think the two rules I want are:

$$u \doteq H \cdot \hat{S}(u[\xi]) \mapsto u \doteq H \cdot \hat{S}(\_)$$

$$u \doteq \hat{R}(u[\sigma]) \mapsto \bot \qquad (\hat{R} \text{ is strongly rigid})$$

2008.3.5

Reading through Twelf's unify.fun. Thoughts:

- What are LVars? Something to do with blocks.
- What are AVars? Haven't a clue.
- Might want invert to pass along a variable that says whether it's in a rigid or nonrigid position.
- Or is this already the distinction between invert and prune? Seems so: prune is rigid, invert is not necessarily.
- pruneSub comment by invertSub???

- What's the deal with waking up constraints? What else should be done about it?

Okay, so Twelf does actually throw the occurs-check when it shouldn't. Consider:

```
o : type.
k : o.
eq : ((o -> o) -> o) -> ((o -> o) -> o) -> type.
refl : eq M M.

% c :     eq ([x] U x) ([x] x (U ([y] k)))
%      -> type.
% test : c refl.

c : {U : (o -> o) -> o} eq ([x] U x) ([x] x (U ([y] k)))
      -> type.
test : c ([x] x k) refl.
```

The commented-out code will crash and burn with an occurs-check, but the code below it works fine. This is because when trying to solve

$$\lambda x.u[x] \doteq x \cdot u[\lambda y.k]$$

Twelf thinks that u on the right is a 'rigid enough' position even thought the substitution isn't a pattern.

2008.3.6

The troubling thing about the way constraints are stored is that both having extra variables without constraints constraining them, and extra constraints without variables witnessing them, seem to be possible errors.

2008.3.8

It seems reasonable to store even singleton MIDI events as durationed sequences, so that operators like "speed up" and "transpose" and so on work uniformly on sequences and singletons.

2008.3.9

Negationless logic doesn't yet make much sense to me. It isees like it's strictly less expressive than inuitionistic logic.

2008.3.10

Watching a John Baez video talking about lattices and Lie Groups and Dynkin diagrams. It seems that he's claiming D_4 is the four-tuples that are all integers or all half-integers, and also that D_n is the n-tuples that sum to an even number. I don't see why these are isomorphic.

The E_8 lattice is supposed to be 8-tuples that are all integers or all half-integers, which also must sum to an even number.

2008.3.12

Intrinsic encoding of LF breaks down when you get to Π typing.

```
tp : type. %name tp T.
ltp : type. %name tp LT.
tm : tp -> type. %name tm M.
ltm : ltp -> type. %name tm LM.
var : ltp -> type. %name var H.
sp : ltp -> tp -> type. %name sp S.

subst/ltp : ltm A -> (var A -> ltp) -> ltp -> type.
subst/tp : ltm A -> (var A -> tp) -> tp -> type.

tt : tp.
fam : tm tt -> tp.

base : tp -> ltp.
pi : {x:ltp} (var x -> ltp) -> ltp.

root : tm T -> ltm (base T).
lam : ({x:var A} ltm (B x)) -> ltm (pi A B).

app : var A -> sp A T -> tm T.

nil : sp (base T) T.
cons : sp B' T -> subst/ltp M B B' -> sp (pi A B) T.

subst/ltp/base : subst/ltp M ([x] base (T x)) (base T')
  <- subst/tp M T T'.
subst/ltp/pi : ???
```

2008.3.13

Consider coverage checking.

A coverage goal is something like $\Gamma \vdash A : \text{type}$. For example, $x : \text{nat}, y : \text{nat} \vdash \texttt{plus}\ x\ y\ u[] : \text{type}$. The question is, how can I use constants from the signature to match something of type A? My signature in the above case probably looks like

```
nat : type. z : nat. s : nat -> nat.
plus : nat -> nat -> nat -> type.
```

```
plus/z : plus z N N.
plus/s : plus (s N) M (s P)
         <- plus N M P.
```

The inputs x and y are 'hard', so they don't unify with $s\ n[]$ or with z. I would have thought this were matching but for the evar $u[]$ coming from the output. Do we just ignore the output then? Anyway, splitting the free variable x leads to two coverage goals:

$$x' : \text{nat}, y : \text{nat} \vdash \texttt{plus}\ (s\ x')\ y\ u[] : \text{type}$$

$$y : \text{nat} \vdash \texttt{plus}\ z\ y\ u[] : \text{type}$$

Assuming the evars in the clauses are given permission to depend on the x', y, these are now immediately covered. But how did we even do splitting? We tried to unify the output types of clauses against nat, and then abstracted. For instance, with s we fully applied it to maximally general evars of the types of its arguments, and then unified its result type with nat, which left no constraints, and generated the evar x'. Hmm. Say $\gg$ for coverage:

$$\frac{c : \Pi\Psi.b \in \Sigma \qquad \theta b = a}{\Sigma \gg (\Gamma \vdash a : \text{type})}$$

$$\frac{\Sigma \sim B = (\Delta_i, \theta_i) \qquad \forall i. \Sigma \gg (\Delta_i, \Gamma \vdash \theta_i a : \text{type})}{\Sigma \gg (\Gamma, \psi : B \vdash a : \text{type})}$$

$$(\Sigma, c : a) \sim B = (\Sigma \sim B), (a \sim B)$$

$$\frac{\exists \Psi. a \doteq b \mapsto \Delta \vdash \theta}{\Pi\Psi. a \sim b = (\Delta, \theta)}$$

My brain is giving up at the notion of how to split on higher types.

2008.3.14

Splitting at higher types doesn't seem quite so crazy to me anymore. You just keep a context of parameters that are introduced by arguments to free variables. These are also available for constructing things at the eventual return type.

Conjecture If a logic program covers with each of the blocks in its regular world added to the signature somehow (not clear what to do with the SOME parts: maybe turn them into free variables?) then it covers over the actual regular world.

$$\frac{c : \Pi\Psi.b \in \Sigma \qquad [\theta/\Psi]b = a}{\Sigma \gg (\Gamma \vdash a : \text{type})}$$

$$\frac{\Sigma; \Psi \sim b = (\Delta_i, \theta_i, R_i) \qquad \forall i. \Sigma \gg (\Delta_i, \Gamma \vdash \theta_i[\lambda\hat{\Psi}.R/\psi]a : \text{type})}{\Sigma \gg (\Gamma, \psi : \Pi\Psi.b \vdash a : \text{type})}$$

$$(\Sigma, c : A); \cdot \sim b = (\Sigma; \cdot \sim b), (c : A \sim b)$$

$$\Sigma; \Gamma, x : A \sim b = (\Sigma \sim b), (x : A \sim b)$$

$$\frac{\exists\Psi.a \doteq b \mapsto \Delta \vdash \theta}{H : \Pi\Psi.a \sim b = (\Delta, \theta, H[\mathsf{id}_\Psi])}$$

Could these specifications of splitting yield coverage results even for LF as self-encoded to yield base-type polymorphism? Eh. Probably not for higher-order programs. Maybe for polymorphic ones.

2008.3.15

The notion I had that suitably structured plain ol' simple graphs (maybe with vertices or edges colored) could serve as a useful starting point for a definition of weak ω-categories isn't really panning out that well.

2008.3.16

One could derive a chain complex from an n-category C in a sort-of-obvious way by taking C_n to be the free Abelian group over the n-cells of C, and let the boundary operator take for instance $f : A \to B$ to $B - A$. Obviously then higher cells would satisfy the chain condition $\partial\partial = 0$, because their codomains and domains would have the same boundary. But this would seem to lose information, as all endomorphisms would have boundary zero.

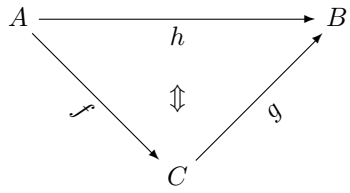
The *geometric* contribution of category theory is a more sophisticated notion of boundary, where a cycle isn't just a cycle, but a *situated* one.

2008.3.17

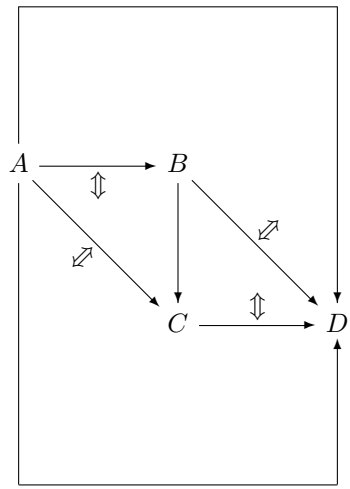
Algebraizing the interchange properties of 'horizontal' composition of higher cells seems very fragile.

I noticed that associativity for composition of 1-cells sort of falls out of the *ability* to compose 2-cells appropriately. If we take the *fact* that

$g \circ f = h$ to be (at least) the existence of a diagram



then we can chase



to see that the any two arrows that are respectively composites of three arrows in the two different orders available are still connected by a two-cell.

So presumably what I want to say is that *weak equality* of two n-paths is the existence of a pair of $n+1$-cells α, β between them such that the path $\alpha;\beta$ is weakly equal to the identity path, and so too $\beta;\alpha$ is weakly equal to the other identity path. This is a sort of coinductive definition, although if you're talking about n-categories for finite n it bottoms out ('tops' out?) at the nth level.

The axiom that (if everything works out right, which it probably won't) yields both the *existence* of composites and their associativity is merely that every path is weakly equal to a *direct* one from its domain to its codomain.

* * * * * *

I think I could avoid strengthening lemmas in the proof of linear cut elimination if I had first-class world-to-world functions.

2008.3.18

Still wishing for a simpler definition of just the graph-theoretic portion of my random thinking about ω-categories.

The essential thing, I thought, was the definition of connected n-paths. The data of a graph is a collection of $(n+1)$-cells that have connected n-path domain and codomain. I staged the definition by putting off the notion of 'connected'. Maybe it would be easier to define *only* the connected n-paths and quotient out by graph identity later? Let's see.

Morphisms between n graphs should be assignments of cells to cells that preserve dom and cod.

Here are the ways to build (connected, but we'll omit that word) $(n+1)$-paths $f : A \to_n B$ from n-path A to n-path B. (The 0-paths are just the 0-cells.)

If f is an $(n+1)$-cell from A to B, then f is an n-path.

If $f : A \to_n B, g : B' \to_n C$, and $\bar{B} \to G \cong \bar{B}' \to G$, then $g \circ_0 f : A \to_n C$. See below about the operation $\bar{X}$.

If $f : A \to_n B, g : C \to_n D$, then $g \circ_{m+1} f : (C \circ_m A) \to_n (D \circ_m B)$ if both the inner composites make sense.

If f is an n-path, then $id_f : f \to_{n+1} f$.

* * * * * *

If f is an n-path $A \to B$ in G, then I can compute from it a generalized subgraph $\bar{f} \to G$ and also show how its domain and codomain are contained in it via $\bar{A} \to \bar{f}$ and $\bar{B} \to \bar{f}$ that commute in the obvious way.

For singleton paths, take a point at the n-cell, and its dom and cod.

For composites, take colimits.

For identity paths, take the dom/cod.

2008.3.22

I think things become easier if I first consider 'untangled paths' that never duplicate path elements. I feel that a general path will always be a homomorphic image of such a special path.

So let a pregraph be a graded collection of cell sets C_n with boundary maps $\partial_{in} : C_{n+1} \to \Pi_n$ with $i \in \{0, 1\}$. The type Π_n consists of triples (s, C, t) where $s, t \in \Pi_{n-1}$ and $C \subseteq C_n$. By convention $\Pi_{-1} = \{*\}$. We conflate (s, C, t) with C for the use of $\in$.

The *sphere* of $(s, C, t) \in \Pi_n$ in a pregraph is the pregraph obtained by keeping only the cells that are hereditarily the domain or codomain of (s, C, t) and adding one more n-cell with $\partial_0(*) = t$ and $\partial_1(*) = s$.

A pregraph is *n-balanced* if for every $x \in C_{n-1}$ there is a unique $y_0 \in C$ such that $x \in \partial_0 y_0$, and a unique $y_1 \in C$ such that $x \in \partial_1 y_1$. An n-cell is *linked* to an $(n-1)$-cell g if g is in the domain or codomain of f. of g that is in the codomain of f. A pregraph is *connected* if the equivalence relation induced by the linked-to relation has only one equivalence class.

$(s, C, t), (s', C', t') \in \Pi_n$ are *compatible* iff $s = s'$ and $t = t'$. $(s, C, t) \in \Pi_n$ is an n-path if its sphere is n-balanced and connected, and s and t are

compatible $(n-1)$-paths.

The unique element of Π_{-1} is a (-1)-path, and considered to be compatible with itself.

A pregraph is a *graph* if ∂ always outputs compatible pairs of n-paths.

* * * * * *

This sphere intuition is seductive. Suppose we have $\mathrm{dom}, \mathrm{cod} : C_{n+1} \to \mathcal{P}C_n$ and $k : C_{n+1} \to C_n$. We demand:

- $\mathrm{dom}\, x \cup \{kx\}$ is n-balanced and connected.
- $\mathrm{cod}\, x \cup \{kx\}$ is n-balanced and connected.

We could ask that the image of $\mathrm{dom}, \mathrm{cod}$ on the one hand ('real cells') is disjoint from k on the other ('virtual cells').

In this case $\mathrm{dom}, \mathrm{cod} : C_{n+1} \to \mathcal{P}R_n$ and $k : C_{n+1} \to V_n$ and $C_n = R_n \cup V_n$.

Actually V_n should be *determined*, shouldn't it, by what is possible at each stage? If $x, y \in \mathcal{P}R_n$ and $z \in V_n$ are such that $x \cup \{z\}$ and $y \cup \{z\}$ are n-balanced and connected, then put an element in V_{n+1} for that triple.

Yes, so: We have a choice of R_n for each n. We must also choose $\mathrm{dom}_i : R_{n+1} \to \mathcal{P}R_n$ and $K : R_{n+1} \to V_n$ where $C_n = R_n \cup V_n$. The V_n are determined as follows: $V_0 = \{*\}$. $V_{n+1} = \{(d_0, d_1, k) \,|\, S(d_i \cup \{k\})\}$ and we can extend dom_i, K to all cells by $\mathrm{dom}_i(d_0, d_1, k) = d_i$ and $K(d_0, d_1, k) = k$. We require $S(\mathrm{dom}_i\ x \cup \{K\ x\})$ for all $x \in R_n$.

Now we define the predicate $S(C)$ for $C \subseteq C_n$. Take all of C's real and virutal domains and codomains hereditarily and we have a new pregraph at least. S holds if this pregraph is n-balanced (every n-cell, most of them real but one virtual, covers every $(n-1)$-cell exactly once by domain and codomain) and connected, by equivalence closure of linkage between real cells.

Try again:

Choose R_n for each n, and $\partial : R_n \to \prod_{i=0}^{n-1} \mathcal{P}R_i \times \mathcal{P}R_i$. Abbreviate $V_n = \prod_{i=0}^{n-1} \mathcal{P}R_i \times \mathcal{P}R_i$. Let $C + k$ be the hereditary restriction of the graph to C with an additional n-cell $*$ that ∂ maps to k. Being n-balanced means every $(n-1)$-cell is (separately) the domain and codomain of a unique n-cell. Being connected means that the equivalence relation arising from δ_0 and δ_1 induces a single equivalence class.

2008.3.23

Balancedness is not really sufficient. Consider popping a two-cell out of a two-sphere that is itself not from the source object to the target object.

2008.3.24

I think a progressive notion of connectedness is appropriate, as reluctant as I am to admit it.

Consider an $\mathbb{N}$-indexed family of sets of 'cells' C_n and maps $\partial_n : C_{n+1} \to \mathcal{P}C_n \times \mathcal{P}C_n$ and $v_n : C_n \to 2$. The functions ∂_{0n} and ∂_{1n} are defined to pick out the first and second projections of the output of ∂. A cell $x \in C_n$ is called 'virtual' if $v(r) = 1$, and 'real' if $v(x) = 0$. We sometimes leave off subscript ns when clear from context.

A set $C \subseteq C_n$ of n-cells is an *n-precycle* if there is exactly one virtual $x \in C$. We write $k(C)$ for this unique virtual cell, and $C^* = C \setminus k(C)$. A 0-precycle C is a 0-*cycle* if its cardinality $|C|$ is 2. An $(n+1)$-precycle C is an $(n+1)$-*cycle* if there are maps $e : |C^*| \to C^*$, $v : |C^*| + 1 \to \mathcal{P}C_n$, $d_0, d_1 : C_n \to C$ such that

- $v(0) = \partial_1 k(C)$
- $v(|C^*|) = \partial_0 k(C)$
- $\forall i \in |C^*|.v(i+1) \cup \partial_0(e(i)) = v(i) \cup \partial_1(e(i))$

("C is connected") and

$$\forall i \in \{0,1\}.\forall x \in C.\forall y \in \partial_i x.d_i(y) = x$$

("C is balanced")

A pair $(D, C) \in \mathcal{P}C_n \times \mathcal{P}C_n$ is an *n-span* if C and D are both n-cycles and $k(C) = k(D)$.

A structure (C_n, ∂, v) is a *flat ω-graph* if

- $\partial_n x$ is a span for every n and $x \in C_{n+1}$.
- For each n span (C, D), there is exactly one virtual cell $x \in C_{n+1}$ such that $\partial x = (C, D)$.

An *ω-graph* is a flat ω-graph together with an equivalence relations $\equiv_n$ on each C_n as well as a family of functions $\{d_{in}^{xy} : \partial_i x \to \partial_i y \,|\, x \equiv_n y \in C_{n+1}\}$ such that $d_i^{xz} = d_i^{yz} \circ d_i^{xy}$ (when all three of x, y, z are equivalent) and $d_i^{xx} = \mathsf{id}$.

Then an ω-category would be an ω-graph with extra connectedness requirements.

Somehow the mere balance criterion seems intriguing, even though I know it isn't sufficient to rule out *at least* two kinds of exotic paths. One would be like a 1-path that has a disconnected cycle elsewhere, and another

one is e.g. a 2-path whose virtual cell "isn't big enough" and so progressing around the sphere has to "borrow" executions of 1-cells, precisely what the "progressive" definition of connectedness rules out.

I think I can neutralize the problems of extra cycles later on in the definition by saying that the output of composition of an n-path and an n-path plus an n-cycle must be the same.

2008.3.25

Wait a minute — all this business with paths and stuff looks just like a sort of free *strict* n-category, since associativity is on the nose here.

So why not say: a weak n-category is a family of distinguished subsets W_n of the cells C_n of a strict n-category, which satisfy the property that for every $x \in C_n$, there is an equivalent $y \in W_n$, where equivalence of $x, y \in C_n$ means there are $f, g \in C_{n+1}$ such that $f \circ g$ and $g \circ f$ are equivalent to identities.

2008.3.26

Is there any nice way of classifying the harmonic functions on subsets of infinite lattices?

I don't think I even need to identify a subset of a strict ω-category to achieve the descriptive benefits I expect to get — so here I am most close to believing that I've simply made an error and fooled myself into thinking I don't need to be particularly sophisticated about coherence.

But I sure would like to know the error I'm making, if any, by simply saying that the data required to specify a weak ω-category *are* the cells of a strict ω-category by shifting one's perspective on what composition means — that a cell 'is a composite' of a pair of cells if it is equivalent in the sense generated by the two rules

$$\frac{f : A \to B : g \qquad fg \equiv \mathsf{id} \qquad gf \equiv \mathsf{id}}{A \equiv B} \qquad \frac{}{A \equiv A}$$

to the actual composite in the strict ω-category, which effectively just represents paths.

2008.3.27

Maybe the subset perspective is useful to eliminate even the possibility of canonical compositions, though?

2008.3.28

Need to collect counterexamples from TL38. One in particular that I want to think about is the one where (weak) pattern inversion is unsound,

since there's a performance worry that pushing through those substitutions all the way might be expensive.

2008.3.29

A V-category where V is a monoidal category is a set X of objects and V-objects $h(x,y)$ for $x,y \in X$, and arrows $\circ : h(y,z) \otimes h(x,y) \to h(x,z)$ and $\mathsf{id} : I \to h(x,x)$.

This seems kind of like saying: take a bicategory B, each of whose homsets resembles V. Choose a mapping k from X to objects of B. For every pair of objects, choose a distinguished 1-cell $h_{xy} : kx \to ky$. Demand that we have a "composition 2-cell" for any $f : kx \to ky$ that goes $f \Rightarrow h_{xy}$.

Or, realizing that a monoidal category should be a bicategory where every object is *equivalent* and not necessarily identical, just say an enriched category is a bicategory B with $h_{xy} : x \to y$ and composition 2-cells $f \Rightarrow h_{xy}$ for each $f : x \to y$. B is then 'locally enriched' over whatever the homcategories happen to be. Not sure how best to impose associativity.

Maybe it falls out of saying: an enriched category is a bicategory where every homcategory has a terminal object, and maybe horizontal composition has to preserve that terminal object. Then again maybe not - the set of (object of) paths over morphisms is not necessarily isomorphic to the direct morphisms. Nah, even terminality seems highly suspect.

2008.3.30

A locally enriched category is a *subbicategory* of a bicategory, where every homset has a terminal object. It is only required to be (and almost certainly will only be) terminal from the perspective of the subcategory of the homset.

A category enriched over a monoidal category V is such a thing where every object is equivalent, and every homset (prior to considering the distinguished subcategory of that homset) looks like V.

Better still: a locally enriched category is a bicategory where every homset has a terminal object. A category enriched over a monoidal category V is a bifunctor from a locally enriched category into V considered as a bicategory with one object. But really we would just enrich over a bicategory!

Consider specifically the bicategory whose objects are some set X, whose 1-cells are strings over X, and whose two-cells $s_1 \to s_2$ is an injective order-preserving map showing how s_2 is a subsequence of s_1. Composition of $s_1 : a \to b$ and $s_2 : b \to c$ is given by 'mediated concatenation' $s_1 b s_2$. The other compositions should be obvious. The local terminal object is the empty string. Maps from this to a monoidal category V seem to look like

categories enriched over V also.

I am not totally certain about this, but it looks like the obvious sort of notion of arrow from one (locally-terminal bicategory over V) to another actually does express the appropriate notion of preservation of composition. Indeed the bicategory doesn't really *need* to be locally-terminal for this definition to make sense.

The weird thing is that in order to define the bicategorical structure on categories enriched over V, I seem to need that V only has one object.

So let a monoidal category V be given. A V-category is a bicategory B and a bifunctor $f : B \to V$. A V-functor $(B_1, f_1) \to (B_2, f_2)$ is a bifunctor $B_1 \to B_2$ that makes the obvious triangle commute. The category of V-categories has a monoidal structure. We take the cells of $B_1 \otimes B_2$ to be pairs of objects from B_1 and B_2, and $f_1 \otimes f_2$ is computed pointwise — we take advantage of the fact that V has only one object to punt on the question of how we 'compose' objects. Maybe the whole construction would work if we are trying to build a less-monoidal category from a more-monoidal one.

2008.4.1

Making glue stretchability a torsor found a bug in my pasting-diagram layout code in a nice way.

2008.4.2

Even a pasting-diagram interpretation of the free braided monoidal category seems really hard to formalize.

Maybe if the 1-cells were also essentially 2-dimensional?

2008.4.3

Consider taking the free category *of* a category: there are more paths than used to be arrows. One gets what I'd expect to be an accurate 'weak categorical' picture if you import 2-cells corresponding to composition facts from the original category, but I'm not sure if this is sufficient.

2008.4.4

How can one make 'micro-code-reuse' easier? How about the same for proofs? The trouble is the background setup required for various functions. Not everybody agrees on the canonical way to define groups, so their proofs are slightly different. On the other hand, there come to be agreements about simple (and I only mean simple in a subjective way) types, so that little combinators having to do with like option and list and stuff are pretty

transportable, because we all agree that option and list are standard useful things.

2008.4.5

I want to ask, 'what is mathematics?' but I know there's no real *is* there.

Mathematics is a word, and its denotation is a big hazy high-dimensional blob that joins up with computer science at one end, and physics at another, and economics at yet a third, and poetry and sheep-counting and accounting and surveying and astronomy at various others.

Asking whether one particular activity 'really is' mathematical or not is to be a european diplomat quibbling over whether Alsace-Lorraine really belongs to France or Germany or what-have-you.

On the other hand, we can all basically agree that Paris is in France.

And while mathematics has no universally agreed-upon center either, no more than it has certain boundaries, there are some themes that find themselves smeared over the rough center of what we all basically agree is definitely mathematics.

I should like to think more about these: formality, abstraction, the fidelity and precision of communicating of mathematical ideas, logic, quantification, proof.

2008.4.6

On bias: imagine that there is an election among population P between options 1 and 2. If 1 is chosen, some subset $D_1 \subseteq P$ are penalized 100 utils. Similarly if 2 is chosen, $D_2 \subseteq P$ are penalized 100 utils. There are agents a_1, a_2 that respectively *receive* 1000 utils if $1, 2$ is chosen. Consider the following cases:

- only a_1 knows D_1, only a_2 knows D_2
- only a_1 knows D_2, only a_2 knows D_1
- everyone knows D_1, D_2

2008.4.7

Is there a reasonable notion of category with arrows violating stratification of dimensions?

I imagine something along the following lines. We're given a set C of cells and an action of the monoid of binary strings $\partial_v : C \to C$ (where $v \in 2^n$) and an involution $—^\perp$ such that $\partial_v x^\perp = (\partial_{\bar{v}} x)^\perp$.

If for all v with $|v| = n$ we have $\partial_{1\bar{v}} f^\perp = \partial_{1v} g$, then n-horizontal composition $f *_n g$ is well-defined and has boundary

$$((\partial_0 g)^\perp *_{n-1} (\partial_0 f)^\perp)^\perp \to \partial_1 f *_{n-1} \partial_1 g$$

Why is $\partial_1 f *_{n-1} \partial_1 g$ well-defined? It's because for all $|v| = n-1$ we have $(\partial_{0v1} f)^\perp = \partial_{1v1} g$. How about $(\partial_0 g)^\perp *_{n-1} (\partial_0 f)^\perp$? It's because for all $|v| = n-1$ we have $(\partial_{0v}(\partial_0 g)^\perp)^\perp = \partial_{1v}(\partial_0 f)^\perp$ because

$$\partial_{1\bar{v}0} g = (\partial_{0\bar{v}0} f)^\perp$$

Grr this involution stuff is too tricky still. Let me try the original thought I had that is less uniform.

Still take C and an action $\partial_v : C \to C$ of 2^n on C. Also a map $\mathsf{id} : C \to C$. $*_n$ is an operation taking f and g such that $\partial_{0v} f = \partial_{1v} g$ for all $v \in 2^n$. We require:

$$\partial_i(f_1 *_0 f_0) = \partial_i f_i$$

$$\partial_i(f *_{n+1} g) = \partial_i f *_n \partial_i g$$

$$\partial_i(\mathsf{id}\ x) = x$$

It should be that $(C, *_n, \mathsf{id})$ forms a category for every n... I'm not sure how best to impose the other associativities.

2008.4.8

I need to recall under what conditions one can actually recover the type of a variable in HLF. There was a funny simplification rule that seemed to apply.

2008.4.9

Typographic layout faces the same problem as type inference that it has information that you want to flow bottom-up, and information that you want to flow top-down.

It's even more complicated because horizontal spacing and vertical spacing for a given primitive might flow in different directions.

And for a single primitive, there might be some information flowing up and some flowing down even within in the same direction, but this we actually have encountered before in types: it seems like simple types should always be pushed bottom-up, and refinements on top of them subtly go up and down based on the bidirectional type-checking that we're used to.

In type inference we also have unification which is 'every way at once' just trying to satisfy constraints. The way tom7 talks about his ideal layout algorithm, he seems to take this perspective, that we ought to just write scoring algorithms and worry about algorithms for optimizing them later.

All quantities below are to be non-negative. A pair of a list of half-integers $\vec{h}$ and a list of integers $\vec{z}$ is a *split vector* of size $\vec{d}$ if $(\mathbf{rev}\,\vec{h})@\vec{z} \leq \vec{d}$.

A diagram of size $\vec{d}$ is a map δ from all the split vectors of size $\vec{d}$ to cells such that

$$\partial_i(\delta(\vec{h}, \vec{z})) = \delta(\mathbf{tl}\,\vec{h}, (\mathbf{hd}\,\vec{h} + i/2) :: \vec{z})$$

Where $\partial_{-1} = \mathrm{dom}$ and $\partial_1 = \mathrm{cod}$. Composition is a map from diagrams to cells that is appropriately domained and associative.

Let morphisms between objects be strings of objects; and 2-cells be duals of order-preserving injections among these strings. Composition of 1-cells $s : a \to b$ and $t : b \to c$ is $sbt : a \to c$.

2008.4.11

I would like to think of composition as 'removal of obstructions' but this seems to require bicategorical thinking even to get categories, hardly breaking even.

2008.4.13

Suppose we have P^n_m. The P^n_0 are given, and P^n_{m+1} is a list of P^n_m satisfying certain composition constraints. dom and cod go from P^n_m to P^{n-1}_{n-1}. Still basically $\partial : P^{n+1} \to \mathrm{list}^n(P_n)^2 \ldots$ but I know this doesn't work.

How about I have just C and a compatibility relation $\subseteq \mathrm{list}^n C \times \mathrm{list}^n C$?

1. If ab and ac and db then dc
2. If ba and ca and bd then cd
3. If $a_i b_i$ for all i then $\vec{a}\vec{b}$.

2008.4.14

How do you do measure theory on $\mathbb{Q}^n$ when everything is countable, and therefore presumably of measure zero? It seems like some box-counting limit should still exist.

2008.4.15

Consider the totality of globular sets that come equipped with composition functions. Call this G.

Current (globular) definitions of weak n-category select a subset of this totality as being 'associative enough'

I could also filter G by imposing the restriction that the globular set arises as a subset of the cells of a strict n-category, and the composition functions are pointwise equivalent (in the coinductive sense, and with this notion of equivalence taking place in the strict n-category) to those of the strict n-category's.

Conjecture: these permit exactly the same elements of G.

One direction: Suppose I have a globular set of 0-, 1-, 2-cells that is an 'equicomposition'-closed subset of some strict n-category $\mathbf{C}$. I want to see that this same set of cells with the same compositions is a bicategory. Equivalence of 2-cells is axiomatically identity.

Because of this, (vertical) 2-cell composition in the weak (putative) bicategory is the same as in the strict, so each homset does form a category. Likewise composition $* : \mathbb{B}(B,C) \times \mathbb{B}(A,B) \to \mathbb{B}(A,C)$ is a functor: If I have

$$\alpha_1 : f_1 \to g_1 : A \to B$$
$$\beta_1 : g_1 \to h_1 : A \to B$$
$$\alpha_2 : f_2 \to g_2 : B \to C$$
$$\beta_2 : g_2 \to h_2 : B \to C$$

then $*(\beta_2 \circ \alpha_2, \beta_1 \circ \alpha_1) = *(\beta_2, \beta_1) \circ *(\alpha_2, \alpha_1)$...

oh wait, no. Equivalence of 2-cells is a little more subtle. For horizontal composition, the domain and codomain require mediator cells so that asking for equivalence with the strict 2-category horizontal composition even makes sense.

2008.4.16

A 'choice of analogues' is sufficient to determine a composition I think.

2008.4.17

Specifically:

Let $\mathbb{C}$ be a strict n-category.

A set $\iota : S \to \mathbb{C}$ over the cells of $\mathbb{C}$ equipped with maps $\alpha, \neg : S \to S$, is an *analogy* for $\mathbb{C}$ if

- $\neg\neg x = x$.
- $\mathrm{dom}(\iota\neg x) = \mathrm{cod}(\iota x)$.
- $\mathrm{dom}(\iota\alpha x) = (\iota\neg x) \circ (\iota x)$.
- $\mathrm{cod}(\iota\alpha x) = \mathsf{id}_{\mathrm{dom}(\iota x)}$.

A map $k : \mathbb{C} \to S$ is a *choice of analogues for* $\mathbb{C}$ if $\mathrm{dom}\,\iota(k(x)) = x$ for all $x \in \mathbb{C}$. The *range of an analogy* $\mathrm{rng}\,\iota$ is the set of cells in $\mathbb{C}$ that hereditarily are the codomain of some analogy cell.

We can define 'compositions' on the range of an analogy, which yield analogy points

$$f *_0 g = k(f \circ_0 g)$$
$$f *_{n+1} g = \mathrm{cod}(\iota k(\iota k(\mathrm{dom}\, f *_n \mathrm{dom}\, g) \circ_0$$
$$(f \circ_{n+1} g) \circ_0$$

$$\iota\neg k(\mathrm{dom}\, f *_n \mathrm{dom}\, g)))$$

??? These are not associativity or exchange on the nose, but up to coherent isomorphisms.

Conjecture All weak ω-categories arise in this way.

2008.4.18

A toplogy specifies which sets are open. Consider a ditopological-ish structure that specifies which posets (on subsets of it) are 'compatible'/'acceptable'.

2008.4.19

My favoring of in "Church on simple types, Curry on everything else" superficially resembles LISP hackers' love of sexps. For the definition of simple types isn't entirely formally clear to me yet; thinking about type reconstruction in HLF convinced me that very probably the universal quantifier really does leave a simple trace, and yet generally I believe that *which* base type you're at oughtn't survive. One of the important criteria is just that the 'less typed' thing is still expressive enough to faithfully embed all the data items you're interested in, but already the natural numbers satisfy this.

2008.4.20

From thoughts long ago, I can represent complexes of polygons in 2-categorical fashion very easily. The missing bit seems only to make the interpretation of a vector biadjoint to its reverse, so that I don't have to think carefully about the progression of time.

By contrast, to think about a planar 'graph' where it's really the colorings of the cells that matter, I live properly in the Poincaré dual.

2008.4.21

Reading through a textbook, Peter Szekeres's "Mathematical Physics", am concerned about the fact that the canonical commutation relation

$$[A, B] = i\hbar I$$

can only be true in an infinite-dimensional Hilbert space. It's sort of blatantly obvious, since the trace on the left would be zero and $i\hbar k$ on the right for a k-dimensional space.

2008.4.22

Let me try to be less tricky about this.

Let a strict n-category be given. Choose a subset W of its cells, with the property that W is closed under domain and codomain.

A set $[\text{—}] : S \to \mathbb{C}$ over the cells of $\mathbb{C}$ equipped with maps $\alpha, \neg : S \to S$, is an *analogy* for $\mathbb{C}$ if

- $\neg\neg x = x$.
- $[\alpha x] : [\neg x] \circ [x] \to \mathsf{id}_{\mathrm{dom}[x]}$.

Let $\mathbb{C}'$ be the cells of $\mathbb{C}$ whose domain and codomain are both in W, including without restriction all objects of $\mathbb{C}$. A *choice of analogues* is a map $k : \mathbb{C}' \to S$ such that $\mathrm{dom}[kx] = x$ and $\mathrm{cod}[kx] \in W$.

This choice of analogues gives rise to composition operators on W. Let $u_n = w \circ_n v$.

$$\begin{array}{lcl}
w *_0 v & = & \mathrm{cod}[ku_0] \\
w *_1 v & = & [k(\mathrm{cod}\, u_1)] \circ_0 \\
 & & \quad \mathrm{cod}[ku_1] \circ_0 \\
 & & [\neg k(\mathrm{dom}\, u_1)] \\
w *_2 v & = & [k(\mathrm{cod}\, u_2)] \circ_0 \\
 & & \quad (\mathsf{id}_{[k(\mathrm{cod}^2\, u_2)]} \circ_1 \\
 & & \quad\quad \mathrm{cod}[ku_2] \circ_1 \\
 & & \quad\quad \mathsf{id}_{[\neg k(\mathrm{dom}^2\, u_2)]}) \circ_0 \\
 & & [\neg k(\mathrm{dom}\, u_2)]
\end{array}$$

This suggests

$$w *_n v = M_n^0$$

$$M_n^m = \begin{cases} \mathsf{id}^m([k(\mathrm{cod}^{m+1}\, u_n)]) \circ_m & \\ \quad M_n^{m+1} \circ_m & \text{if } m < n; \\ \quad \mathsf{id}^m([\neg k(\mathrm{dom}^{m+1}\, u_n)]) & \\ \mathrm{cod}[ku_n] & \text{if } m = n \end{cases}$$

No, this isn't quite right. I don't feel good about $[k(\mathrm{cod}\, u_2)]$ in the definition of $w *_2 v$. It should be $[k(\mathrm{cod}\, w *_1 \mathrm{cod}\, v)]$.

I think I'd rather define $*$ to give the whole mediating cell inductively and take its cod after the fact.

Also I don't seem to use α intrinsically.

Just take $W, U \subseteq \mathbb{C}$, mappings $\text{—}^{-1}, \alpha : U \to U$ with $\alpha : x^{-1}x \to \mathsf{id}_{\mathrm{dom}\, x}$ and $(x^{-1})^{-1} = x$. Let $\mathbb{C}'$ be as before $\{x \in \mathbb{C} \mid \mathrm{dom}\, x \in W \wedge \mathrm{cod}\, x \in W\}$ and suppose we have $k : \mathbb{C}' \to U$ such that $\mathrm{dom}(kx) = x$ and $\mathrm{cod}(kx) \in W$. Define precomposition by

$$w *_n v = kE_n^0$$

$$C_n^m = \mathrm{cod}^m\, w *_{n-m} \mathrm{cod}^m\, v$$

$$D_n^m = \mathrm{dom}^m\, w *_{n-m} \mathrm{dom}^m\, v$$

$$E_n^m = \begin{cases} \mathsf{id}^m(C_n^{m+1}) \circ_m E_n^{m+1} \circ_m \mathsf{id}^m((D_n^{m+1})^{-1}) & \text{if } m < n; \\ w \circ_n v & \text{if } m = n \end{cases}$$

The compositions in C and D are obviously justified. Claim I:

$$\partial(\mathrm{cod}(w *_{n+1} v)) = \mathrm{cod}(\partial w *_n \partial v)$$

To see that $E_n^{m+1} \circ_m \mathsf{id}^m((D_n^{m+1})^{-1})$ and $\mathsf{id}^m(C_n^{m+1}) \circ_m E_n^{m+1}$ are justified, we want to know

$$\mathrm{dom}(D_n^{m+1}) = \mathrm{dom}^{m+1}\, E_n^{m+1}$$

$$\mathrm{dom}(C_n^{m+1}) = \mathrm{cod}^{m+1}\, E_n^{m+1}$$

or to reduce notation

$$\mathrm{dom}(D_n^m) = \mathrm{dom}^m\, E_n^m$$

$$\mathrm{dom}(C_n^m) = \mathrm{cod}^m\, E_n^m$$

for $m \geq 1$. It's easy to check for $n = m$. Suppose $n > m$.

$$\mathrm{dom}(f *_n g) = E_n^0$$

Observe also

$$D_n^m(\mathrm{dom}^j\, w, \mathrm{dom}^j\, v) = D_{m+j}^{n+j}(w, v)$$

$$C_n^m(\mathrm{dom}^j\, w, \mathrm{dom}^j\, v) = C_{m+j}^{n+j}(w, v)$$

$$E_n^m(\mathrm{dom}^j\, w, \mathrm{dom}^j\, v) =?$$

$$\mathrm{dom}(D_n^m) = \mathrm{dom}(\mathrm{dom}^m\, w *_{n-m} \mathrm{dom}^m\, v)$$

$$\mathrm{dom}^m\, E_n^m = \mathrm{dom}^m(\mathsf{id}^m(C_n^{m+1}) \circ_m E_n^{m+1} \circ_m \mathsf{id}^m((D_n^{m+1})^{-1}))$$

$$= C_n^{m+1} \circ_0 \mathrm{dom}^m\, E_n^{m+1} \circ_0 (D_n^{m+1})^{-1}$$

$$= C_n^{m+1} \circ_0 D_n^{m+1} \circ_0 (D_n^{m+1})^{-1}$$

I've figured it out now. Let ∂ stand for dom or cod and Δ for C or D.

First show

$$\Delta^m_n(\partial' a, \partial' b) = \Delta^{m+1}_{n+1}(a, b)$$

which is pretty easy. Using this in the step case $m < n$, we can see

$$E^m_n(\partial a, \partial b) = \partial E^{m+1}_{n+1}(a, b)$$

The base case $m = n$ is just the property of boundaries we expect for $\circ_{n+1}$. Iterate this to get

$$E^0_{n-m}(\partial^m a, \partial^m b) = \partial^m E^m_n(a, b)$$

the left side of which is also $\mathrm{dom}(\Delta^m_n)$. And this is what we need to license the two compositions in the definition.

What we also want is $\partial(a \odot_{n+1} b) = \partial a \odot_n \partial b$ where $a \odot_n b = E^0_n(a, b)$. But this follows by

$$\begin{aligned}
\partial(a \odot_{n+1} b) &= \partial(E^0_{n+1}(a, b)) \\
&= \partial(C^1_{n+1}(a, b) \circ_0 E^1_{n+1}(a, b) \circ_0 (D^1_{n+1}(a, b))^{-1}) \\
&= \mathrm{cod}(\Delta^1_{n+1}(a, b)) \\
&= \mathrm{cod}(\partial a *_n \partial b) \\
&= \mathrm{cod}(kE^0_n(\partial a, \partial b)) \\
&= E^0_n(\partial a, \partial b) \\
&= \partial a \odot_n \partial b
\end{aligned}$$

Once again, the definition of everything:

Let $W, U \subseteq \mathbb{C}$, mappings $—^{-1}, \alpha : U \to U$ with $\alpha : x^{-1}x \to \mathsf{id}_{\mathrm{dom}\, x}$ and $(x^{-1})^{-1} = x$. Let $\mathbb{C}'$ be as before $\{x \in \mathbb{C} \mid \mathrm{dom}\, x \in W \wedge \mathrm{cod}\, x \in W\}$ and suppose we have $k : \mathbb{C}' \to U$ such that $\mathrm{dom}(kx) = x$ and $\mathrm{cod}(kx) \in W$. Define composition by

$$\Delta^m_n(a, b) = k(\partial^m a \odot^0_{n-m} \partial^m b)$$

$$a \odot^m_n b =$$

$$\begin{cases} \mathsf{id}^m(C^{m+1}_n(a, b)) \circ_m (a \odot^{m+1}_n b) \circ_m \mathsf{id}^m(D^{m+1}_n(a, b)^{-1}) & \text{if } m < n; \\ a \circ_n b & \text{if } m = n \end{cases}$$

2008.4.23

Having an involutive $—^{-1}$ on elements of the strict ω-category doesn't seem to be right; consider a category with $f, g : A \to B$ and $h : B \to A$ and

$F, G : A \to A$ where compositions are determined by the *earliest* nonidentity arrow. Then f and g both compete for h has their inverse.

Maybe I could relax involutivity to Brouwer's theorem, but certainly I can get by with defining twist and α on bicomposable pairs.

2008.4.24

Involutivity doesn't seem to matter after all for determining the status of a single cell as equivalence or not; I can coalesce inverse pairs.

2008.4.24

Something I didn't appreciate about Lie algebras before: if you try to compute products of exponentials, you can describe them with the commutators of generators; a fact that looks suspiciously like needing distributive laws for composition of monads. Compare $\lambda : ST \to TS$ with $[T, S] = TS - ST$.

2008.4.25

I actually sort of expect a theory of data dependency to look exactly like a parametrized proof irrelevance:

$$\frac{\Gamma^{\oplus\Phi} \vdash A}{\Gamma \vdash \Diamond_\Phi A} \qquad \frac{\Gamma, A\ \mathsf{poss}_\Phi \vdash C}{\Gamma, \Diamond_\Phi A \vdash C}$$

Where $\Gamma^{\oplus\Phi}$ is defined by

$$(A\ \mathsf{poss}_{\Phi'})^{\oplus\Phi} = A\ \mathsf{poss}_{\Phi' \setminus \Phi}$$

and by convention $\mathsf{poss}_\emptyset = \mathsf{true}$.

2008.4.26

For type operators in the image of the translation of LLF we have the property

If $x : A@\alpha \in \Gamma$ and x appears in M and $\Gamma \vdash M \Leftarrow B[p]$, then $p \geq \alpha$.

2008.4.27

Restart on weak ω-categories for the millionth time.

Let $\mathbb{C}$ and $W \subseteq \mathbb{C}$. Define $\mathbb{C}_W$ to be those cells of $\mathbb{C}$ that have domain and codomain in W. Define $\mathbb{C}^\circ$ to be the set of bicomposable pairs of cells in $\mathbb{C}$. Let $\mathbb{C}^\bullet$ be the maximal subset of $\mathbb{C}^\circ$ closed under swap and $(f, g) \mapsto (fg, \mathsf{id})$.

$(\mathbb{C}, W, k, k^{-1})$ is a weak ω-category if $\mathbb{C}$ is a strict ω-category, $W \subseteq \mathbb{C}$, $k, k^{-1} : \mathbb{C}_W \to \mathbb{C}$ satisfying $\mathrm{dom}(kx) = x$ and $\mathrm{cod}(kx) \in W$ and $(kx, k^{-1}x) \in \mathbb{C}^\bullet$.

Based on this we can define $\Box_n x, \Box_n^{-1} x$, so that $x *_n y = \Box_n(x \circ_n y)$.

The invariant for $\Box_n$ is that dom^{n+1} and cod^{n+1} are already in W.

$$\Box_0 x = kx$$

$$\Box_1 x = \Box_0(k \operatorname{cod} x \circ x \circ k^{-1} \operatorname{dom} x)$$

$$\Box_2 x = \Box_1(kc^2 x \circ_1 x \circ_1 k^{-1} d^2 x)$$

Still not quite right.

A general thought on planning: it is good to have loci of accumulation.
What are mine?
My notebooks.
This file.
My subversion repository.

Some kind of monotonicity should generalize the splitting property of LLF, not usre what.

Enjoyed words from Kubla Khan: incense, revive, mingled, fragments, enfolding, device, honey, sacred.

2008.4.28

A story of a prophecy that fails to come true.

The idea that immersive entertainment is an admixture of creating new *places* and discovering them. Imagine 15th-century European criticizing the New World as *not real*, admiting however that people have been there. Its fault is that it's a *waste of time*.

The role of belief in a deity as the mechanism by which we expect to be altruistically punished even in isolation from other people.

"Ambrose, I've had enough."

2008.4.29

Let there be k_n^d where $d \in \{\triangleleft, \triangleright\}$. The invariant of $\Box_n$ is that the ∂^n are already in W. The output is entirely in W.

$$\Box_0^d x = x$$

$$\Box_{n+1} x = (k^{\triangleright} \Box_n(\operatorname{cod} x) \circ x \circ k^{\triangleleft} \Box_n(\operatorname{dom} x))$$

No, this isn't right either.

It should totally fall out as a unary version of composition. Let me try to do that.

$$\Delta_n^m(a, d) = \mathsf{id}^m(k^d(\Box_{n-m-1}^0 \partial_d^{m+1} a))$$

$$\Box_n^m a = \begin{cases} \Delta_n^m(a, \triangleright) \circ_m (\Box_n^{m+1} a) \circ_m \Delta_n^m(a, \triangleleft) & \text{if } m < n; \\ a & \text{if } m = n \end{cases}$$

Actual transfer into W is given by $x \mapsto \mathrm{cod}(k \Box x)$.

2008.4.30

Apparently it's quite standard to view irreps as orbits. I'm still puzzling over Dolan's description of the orbisimplex.

2008.5.1

Understanding the monotonicity property I want for coverage checking is difficult at higher-order function types, unsurprisingly.

2008.5.2

I think I may have it, now. R_q^p is a predicate on types, with p input and q output. $L_{\vec{p}}^q$ is a predicate on types, with both q and $\vec{p}$ input.

$$\frac{}{R_p^p(a \cdot S)} \qquad \frac{L_\cdot^\epsilon(A) \qquad R_q^p(B)}{R_q^p(A \to B)}$$

$$\frac{R_q^r(A)}{R_q^p(A@r)} \qquad \frac{R_q^p([p/\alpha]A)}{R_q^p(\downarrow\alpha.A)} \qquad \frac{R_q^p(A)}{R_q^p(\forall\alpha.A)}$$

$$\frac{q \leq p_1 \cdots q \leq p_n}{L_{\vec{p}}^q(a \cdot S)} \qquad \frac{R_r^\epsilon(A) \qquad L_{\vec{p},r}^q(B)}{L_{\vec{p}}^q(A \to B)}$$

$$\frac{L_{\vec{p}}^r(A)}{L_{\vec{p}}^q(A@r)} \qquad \frac{L_{\vec{p}}^q([q/\alpha]A)}{L_{\vec{p}}^q(\downarrow\alpha.A)} \qquad \frac{L_{\vec{p}}^q([r/\alpha]A)}{L_{\vec{p}}^q(\forall\alpha.A)}$$

2008.5.3

Do I really need to keep track of a context in the definitions above?

2008.5.4

Is all energy in the universe packaged in discrete multiples of $\hbar$, or is it just any particular system that has a quantized energy spectrum? I can imagine matrices with whatever eigenvalues I like, but maybe the universe is more choosy than that.

2008.5.5

To divide into k sections with margin m, I must have k divides $402 - m(k-1)$, equivalent to $k \mid 402 + m$.

Regarding the system from the 2nd.

The interpretation of $L^q_{\vec{p}}$ is: if you're eliminating the type starting at q, the result will be below everything in $\vec{p}$, and everything is monotone hereditarily. The interpretation of R^p_q is: if you start with p (on the right), you get out q.

Lemma 0.25 *If $A \in LLF$, then $R^p_p(A)$, and $L^p_p(A)$.*

Lemma 0.26 *If $L^q_{\vec{p}}$ and $\vec{r} \subseteq \vec{p}$, then $L^q_{\vec{r}}$.*

Lemma 0.27 *If everything in the context and signature is $L^\epsilon_\cdot$, and some variable $x : A \in \Gamma$ actually occurs in M with $L^\epsilon_{\vec{p}}(A)$, and $R^r_q(B)$, and $\Gamma \vdash M \Leftarrow B[r]$, then $q \geq \vec{p}$.*

2008.5.5

Lemma 0.28 *If $A \in LLF$, and $q \leq p$ then $R^p_q(A)$. If $A \in LLF$, and $q \leq \vec{p}$ then $L^q_{\vec{p}}(A)$.*

Proof of $\multimap$ case:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{i.h.}{L^\alpha_\cdot(A)}}{L^\epsilon_\cdot(A@\alpha)} \quad \dfrac{\dfrac{i.h.}{R^{\alpha*p}_q(B)}}{R^p_q(B@(\alpha*p))}}{R^p_q(A@\alpha \to B@(\alpha*p))}}{R^p_q(\downarrow\beta.A@\alpha \to B@(\alpha*\beta))}}{R^p_q(\forall\alpha.\downarrow\beta.A@\alpha \to B@(\alpha*\beta))}}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{i.h.}{R^X_r(A)}}{R^\epsilon_r(A@X)} \quad \dfrac{\dfrac{i.h.}{L^{X*q}_{\vec{p},r}(B)}}{L^q_{\vec{p},r}(B@(X*q))}}{L^q_{\vec{p}}(A@X \to B@(X*q))}}{L^q_{\vec{p}}(\downarrow\beta.A@X \to B@(X*\beta))}}{L^q_{\vec{p}}(\forall\alpha.\downarrow\beta.A@\alpha \to B@(\alpha*\beta))}}$$

Lemma 0.29 *Suppose every type in the context and signature is $L^\epsilon_\cdot$.*

1. *If a variable $x : A \in \Gamma$ occurs in M with $L^\epsilon_{\vec{p}}(A)$, and $R^r_q(B)$, and $\Gamma \vdash M \Leftarrow B[r]$, then $q \geq \vec{p}$.*
2. *If a variable $x : A \in \Gamma$ occurs in S with $L^\epsilon_{\vec{p}}(A)$, and $\Gamma \vdash S : B[r] > c[q]$, then $q \geq \vec{p}$.*

3. *If* $L^r_{\vec{p}}(A)$ *and* $\Gamma \vdash S : A[r] > c[q]$, *then* $q \geq \vec{p}$.

Proof By induction.

1. Lambda case is easy. We preserve the invariant by inspection.

   Variable case: ...

2.

3.

Remember:
Words are not points.
Economic value is torsorial.
The vast majority of economic value is essentially speculative.

With bounded rationality, the estimation of value should propagate along something like the heat equation, which critically makes sense — because the Laplacian makes sense — even for torsors, even on undirected graphs.

2008.5.6

Testing unification. Running `prop-calc/sources.cfg`. Error message:

```
equiv.elf:13.1-13.38 Error:
Typing ambiguous -- unresolved constraints
A1 = A2 x;
A1 = A2 x;
A3 = A4 x;
A3 = A4 x.
```

Smaller counterexample:

```
o : type.
f : o -> o -> o.
pred : o -> type.
k : pred (f A (f B A)).
pair : pred (f A (f B (f A B))).
mp : pred (f A B) -> pred A -> pred B.
abs : (pred A -> pred B) -> pred (f A B) -> type.
apair : abs ([x] pair) (mp k pair).
```

Smaller still:

```
o : type.
pred : o -> type.
pair : pred A.
abs : (o -> pred A)  -> type.
apair : abs ([x] pair).
```

2008.5.8

$$\frac{\Gamma \vdash s \qquad s * q \geq \overline{p}}{R^p_{q;\Delta}(a \cdot S)} \qquad \frac{L^\epsilon_\cdot(A) \qquad R^p_{q;\Delta}(B)}{R^p_{q;\Delta}(A \to B)} \qquad \frac{R^p_{q;\Delta,\alpha}(A)}{R^p_{q;\Delta}(\forall \alpha.A)}$$

$$\frac{p \geq q}{L^p_q(a \cdot S)} \qquad \frac{R^\epsilon_{r;\cdot}(A) \qquad L^p_q(B) \qquad q \geq r}{L^p_q(A \to B)} \qquad \frac{L^p_q([X/\alpha]A)}{L^p_q(\forall \alpha.A)}$$

Lemma 0.30 *Suppose* $L^\epsilon_*(\Gamma)$

1. *If* $L^\epsilon_r(A)$ *and* $R^p_{q;\Delta}(B)$ *and* $x : A \in M$ *and* $\Gamma \vdash M : B[p]$ *then exists* s *in* Δ *such that* $s * q \geq r$.
2. *If* $L^\epsilon_r(A)$ *and* $x : A \in R$ *and* $\Gamma \vdash R : a[p]$ *then* $\overline{p} \geq r$
3. *If* $L^\epsilon_r(A)$ *and* $x : A \in S$ *and* $L^q_s(B)$ *and* $\Gamma \vdash S : B[q] > a[p]$ *then* $\overline{p} \geq r$.
4. *If* $L^q_r(A)$ *and* $\Gamma \vdash A[q] > a[p]$ *then* $\overline{p} \geq r$.

Proof ■

2008.5.9

What is the role of monotonicity when introducing free variables? Possibly none. Not sure. Probably should confine monotonicity reasoning to splitting rather than let it infect unification.

2008.5.10

Let a set X and a a finitely additive measure μ over X be given. That is, $\mu(\emptyset) = 0$, $\mu(X) = 1$, $\mu(A \uplus B) = \mu A + \mu B$, and $0 \leq \mu A \leq 1$ for all A.

This lets me interpret statements like $p(A \wedge B) + p(\neg C) \geq 1/2$ as $\mu(A \cap B) + \mu(X \setminus C) \geq 1/2$.

Say the syntax is something like

$$\begin{aligned} \text{Propositions } A &::= \top \mid \neg A \mid A_1 \wedge A_2 \\ \text{Arithmetic Expressions } E &::= \mu A \mid r \mid E_1 + E_2 \\ \text{Sentences } S &::= E_1 \geq E_2 \mid E_1 = E_2 \end{aligned}$$

Where r is a real number. Let a theory T, a set of statements, be given. I wish to prove a completeness theorem, like: if $T \vDash S$, then $T \vdash S$. So we're going to take such a T and construct a syntactic model from it. Its points are (classical propositional logic) provability equivalence classes of propositions. If we look at any set X of those, we have to determine a measure for it. It ought to be bounded below by any r such $T \vdash Pr(A) \geq r$ for some A in X, and bounded above by any r such that $T \vdash Pr(A) \leq r$ for some A not in X... No, that doesn't work.

Dang, it really seems that I need to pass to Boolean algebras generally, rather than dealing with concrete powersets.

* * * * * *

I could scale back to saying:
$T \vDash Pr(A) = 1$ iff $T \vdash A$, but this seems too weak to be interesting.

* * * * * *

But just to page it back in, why is classical logic complete for interpretation in powersets?

Let Γ be given. The points of the syntactic model are provability equivalence classes of propositions entailed by Γ, but leave out the one that $\bot$ is in. Interpret each proposition as the set of propositions that (together with Γ) entail it. If a proposition is entailed by Γ, its denotation is the entire set. If its denotation is the entire set, it includes $\top$, and by definition (using cut elimination and the fact that $\top$ is immediately provable) it's entailed by Γ alone. This is as desired.

We must only show that this definition is actually intersection on $\wedge$ and complement on $\neg$.

First: We have $I_\Gamma(A)$ and $I_\Gamma(B)$, the set of consistent propositions that with Γ prove A and B, respectively. We want to confirm $I_\Gamma(A \wedge B)$ is equal to their intersection. Let $C \in I_\Gamma(A \wedge B)$ be given, which yields a derivation $\Gamma, C \vdash A \wedge B$. By inversion $\Gamma, C \vdash A$ and $\Gamma, C \vdash B$. Likewise we can use $\wedge I$ to get from the two derivations to the one.

Second: We have $C \in I_\Gamma(\neg A)$ and wish to show $C \notin I_\Gamma(A)$. This is easy, because we would get a contradiction otherwise. To see the other half, that if $C \notin I_\Gamma(\neg A)$, then $C \in I_\Gamma(A)$, we have to consider two possibilities. One, $\Gamma, C \nvdash \neg A$... Oh. I had to restrict attention to 'worlds' where all propositional atoms are decided from the beginning.

Oh man, maybe that syntactic proof of focussing's correctness via translation into ordered logic works nicely after all.

Make up two atoms $\triangleleft$ and $\triangleright$.

$$A = v^+ F^+ \mid v^- F^-$$

$$F^+ = F^+ \otimes F^+ \mid d^+A$$
$$F^- = F^+ \multimap F^- \mid d^-A$$

Define $\overleftarrow{X}$ and $\overleftarrow{\overleftarrow{A}}$ and $\overrightarrow{X}$ and $\overrightarrow{\overrightarrow{A}}$:

$$\overleftarrow{\overleftarrow{A}} = !(\triangleleft\triangleright \twoheadrightarrow \overleftarrow{A})$$
$$\overrightarrow{\overrightarrow{A}} = \triangleleft\triangleright \bullet !\overrightarrow{A}$$

X	$\overleftarrow{X}$	$\overrightarrow{X}$
v^+F^+	$\triangleleft \bullet \overleftarrow{F^+} \bullet \triangleright$	$\overrightarrow{F^+}$
v^-F^-	$\overleftarrow{F^-}$	$\triangleleft \twoheadrightarrow \overrightarrow{F^-}$
d^+A	$\triangleright \twoheadrightarrow (\triangleright \bullet \overleftarrow{\overleftarrow{A}})$	$\triangleleft\triangleright \twoheadrightarrow \overrightarrow{\overrightarrow{A}}$
d^-A	$\triangleleft\triangleright \bullet \overleftarrow{\overleftarrow{A}}$	$\triangleright \twoheadrightarrow \overrightarrow{\overrightarrow{A}}$
$F_1^+ \otimes F_2^+$	$\triangleright \twoheadrightarrow (\overleftarrow{F_1^+} \bullet \overleftarrow{F_2^+} \bullet \triangleright)$	$\overrightarrow{F_1^+} \bullet \overrightarrow{F_2^+}$
$F^+ \multimap F^-$	$\overrightarrow{F^+} \twoheadrightarrow \overleftarrow{F^-}$	$\overleftarrow{F^+} \twoheadrightarrow \overrightarrow{F_-}$

Now prove:

Lemma 0.31

1. $\overleftarrow{F^+} \bullet \triangleright \dashv\vdash \triangleright \bullet !\overrightarrow{F^+}$
2. $\triangleright \twoheadrightarrow \overleftarrow{F^-} \dashv\vdash \overrightarrow{F^-}$
3. $\overleftarrow{\overleftarrow{A}}; \triangleleft\triangleright \vdash \overrightarrow{\overrightarrow{A}}$
4. $\overrightarrow{\overrightarrow{A}}; \triangleleft\triangleright \vdash \overleftarrow{\overleftarrow{A}}$.

Proof By induction on the proposition.

1. Forward d^+:

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Part 3}}{\overleftarrow{\overleftarrow{A}}; \triangleleft\triangleright \vdash \overrightarrow{\overrightarrow{A}}}}{\overleftarrow{\overleftarrow{A}} \vdash !(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{\overrightarrow{A}})}}{\triangleright \bullet \overleftarrow{\overleftarrow{A}} \vdash \triangleright \bullet !(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{\overrightarrow{A}})}}{\triangleright \twoheadrightarrow (\triangleright \bullet \overleftarrow{\overleftarrow{A}}), \triangleright \vdash \triangleright \bullet !(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{\overrightarrow{A}})}$$

Backward d^+:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Part 4}}{\overrightarrow{A};\triangleleft\triangleright\vdash\overleftarrow{A}}}{\overrightarrow{A}\vdash\overleftarrow{A}}}{\triangleleft\triangleright\twoheadrightarrow\overrightarrow{A};\triangleleft\triangleright\vdash\overleftarrow{A}}}{\triangleleft\triangleright\twoheadrightarrow\overrightarrow{A};\cdot\vdash\overleftarrow{A}}}{\triangleleft\triangleright\twoheadrightarrow\overrightarrow{A};\cdot\vdash\triangleright\twoheadrightarrow(\triangleright\bullet\overleftarrow{A})}}{\overrightarrow{d^+A};\triangleright\vdash\overleftarrow{d^+A}\bullet\triangleright}$$

Forward $\otimes$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\triangleright,!\overrightarrow{F_1^+},!\overrightarrow{F_2^+},\vdash\triangleright\bullet!(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+})}}{\overleftarrow{F_1^+},\triangleright,!\overrightarrow{F_2^+},\vdash\triangleright\bullet!(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+})}}{\overleftarrow{F_1^+},\overleftarrow{F_2^+},\triangleright\vdash\triangleright\bullet!(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+})}}{\triangleright\twoheadrightarrow(\overleftarrow{F_1^+}\bullet\overleftarrow{F_2^+}\bullet\triangleright),\triangleright\vdash\triangleright\bullet!(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+})}$$

Backward $\otimes$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+};\triangleright\vdash\triangleright\bullet!\overrightarrow{F_1^+}\bullet!\overrightarrow{F_2^+}}XXX}{\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+};\triangleright\vdash\overleftarrow{F_1^+}\bullet\triangleright\bullet!\overrightarrow{F_2^+}}}{\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+};\triangleright\vdash\overleftarrow{F_1^+}\bullet\overleftarrow{F_2^+}\bullet\triangleright}}{(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+});\cdot\vdash\triangleright\twoheadrightarrow(\overleftarrow{F_1^+}\bullet\overleftarrow{F_2^+}\bullet\triangleright)}}{(\overrightarrow{F_1^+}\bullet\overrightarrow{F_2^+});\triangleright\vdash(\triangleright\twoheadrightarrow(\overleftarrow{F_1^+}\bullet\overleftarrow{F_2^+}\bullet\triangleright))\bullet\triangleright}$$

2. Forward d^-:

$$\dfrac{\dfrac{\dfrac{\text{Part 3}}{\overleftarrow{A};\triangleleft\triangleright\vdash\overrightarrow{A}}}{(\triangleleft\triangleright\bullet\overleftarrow{A})\vdash\overrightarrow{A}}}{\triangleright\twoheadrightarrow(\triangleleft\triangleright\bullet\overleftarrow{A})\vdash\triangleright\twoheadrightarrow\overrightarrow{A}}$$

Backward d^-:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Part 4}}{\overrightarrow{A};\triangleleft\triangleright\vdash\overleftarrow{A}}}{\overrightarrow{A};\cdot\vdash\overleftarrow{\overleftarrow{A}}}}{\overrightarrow{A};\triangleleft\triangleright\vdash\triangleleft\triangleright\bullet\overleftarrow{\overleftarrow{A}}}}{\overrightarrow{\overrightarrow{A}}\vdash\triangleleft\triangleright\bullet\overleftarrow{\overleftarrow{A}}}}{\triangleright\twoheadrightarrow\overrightarrow{\overrightarrow{A}}\vdash\triangleright\twoheadrightarrow(\triangleleft\triangleright\bullet\overleftarrow{\overleftarrow{A}})}$$

Forward $\multimap$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}},\overrightarrow{\overrightarrow{F^+}}\vdash\overleftarrow{\overleftarrow{F_-}}}{\triangleright\twoheadrightarrow(\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}}),\triangleright,\overrightarrow{\overrightarrow{F^+}}\vdash\overleftarrow{\overleftarrow{F_-}}}}{\triangleright\twoheadrightarrow(\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}}),\overleftarrow{\overleftarrow{F^+}},\triangleright\vdash\overleftarrow{\overleftarrow{F_-}}}}{\triangleright\twoheadrightarrow(\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}}),\overleftarrow{\overleftarrow{F^+}}\vdash\overrightarrow{\overrightarrow{F_-}}}}{\triangleright\twoheadrightarrow(\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}})\vdash\overleftarrow{\overleftarrow{F^+}}\twoheadrightarrow\overrightarrow{\overrightarrow{F_-}}}$$

Backward $\multimap$:

$$\dfrac{\dfrac{\dfrac{\overleftarrow{\overleftarrow{F^+}}\twoheadrightarrow\overrightarrow{\overrightarrow{F_-}},\overleftarrow{\overleftarrow{F^+}}\vdash\overrightarrow{\overrightarrow{F^-}}}{\overleftarrow{\overleftarrow{F^+}}\twoheadrightarrow\overrightarrow{\overrightarrow{F_-}},\overleftarrow{\overleftarrow{F^+}},\triangleright\vdash\overleftarrow{\overleftarrow{F^-}}}\,XXX}{\overleftarrow{\overleftarrow{F^+}}\twoheadrightarrow\overrightarrow{\overrightarrow{F_-}},\triangleright,\overrightarrow{\overrightarrow{F^+}}\vdash\overleftarrow{\overleftarrow{F^-}}}}{\overleftarrow{\overleftarrow{F^+}}\twoheadrightarrow\overrightarrow{\overrightarrow{F_-}}\vdash\triangleright\twoheadrightarrow(\overrightarrow{\overrightarrow{F^+}}\twoheadrightarrow\overleftarrow{\overleftarrow{F^-}})}$$

3. v^+F^+ case:

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Part 1}}{\overrightarrow{\overrightarrow{F^+}},\triangleright\vdash\triangleright\bullet!\overrightarrow{\overrightarrow{F^+}}}\quad\dfrac{\overrightarrow{\triangleleft,\triangleright,!\overrightarrow{\overrightarrow{F^+}}\vdash\triangleleft\triangleright\bullet!\overrightarrow{\overrightarrow{F^+}}}}{\triangleleft,\triangleright,!\overrightarrow{\overrightarrow{F^+}}\vdash\overrightarrow{\overrightarrow{v^+F^+}}}}{\triangleleft,\overleftarrow{\overleftarrow{F^+}},\triangleright\vdash\overrightarrow{\overrightarrow{v^+F^+}}}\,\text{cut}}{\overleftarrow{\overleftarrow{v^+F^+}}\vdash\overrightarrow{\overrightarrow{v^+F^+}}}}{\triangleleft\triangleright\twoheadrightarrow\overleftarrow{\overleftarrow{v^+F^+}},\triangleleft\triangleright\vdash\overrightarrow{\overrightarrow{v^+F^+}}}$$

$v^- F^-$ case:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overleftarrow{F^-} \vdash \overleftarrow{F^-}}}{\overleftarrow{v^- F^-}, \triangleleft, \triangleright \vdash \overleftarrow{F^-}}}{\overleftarrow{v^- F^-}, \triangleleft \vdash \triangleright \twoheadrightarrow \overleftarrow{F^-}} \quad \dfrac{}{\triangleright \twoheadrightarrow \overleftarrow{F^-} \twoheadrightarrow \overrightarrow{F^-}}\ \text{Part 2}}{\overleftarrow{v^- F^-}, \triangleleft \vdash \overrightarrow{F^-}}\ \text{cut}$$

$$\dfrac{\dfrac{}{\triangleleft\triangleright \vdash \triangleleft\triangleright} \qquad \dfrac{\dfrac{}{\overleftarrow{v^- F^-} \vdash \overrightarrow{v^- F^-}}}{\overleftarrow{v^- F^-}; \cdot \vdash !\overrightarrow{v^- F^-}}}{\overleftarrow{v^- F^-}; \triangleleft\triangleright \vdash \triangleleft\triangleright \bullet !\overrightarrow{v^- F^-}}$$

4. $v^+ F^+$ case:

$$\dfrac{\dfrac{}{\triangleleft \vdash \triangleleft} \quad \dfrac{}{\overrightarrow{F^+}; \triangleright \vdash \overleftarrow{F^+} \bullet \triangleright}\ \text{Part 1}}{\overrightarrow{F^+}; \triangleleft\triangleright \vdash \triangleleft \bullet \overleftarrow{F^+} \bullet \triangleright}$$

$$\dfrac{}{\overrightarrow{v^+ F^+}; \triangleleft\triangleright \vdash \overleftarrow{v^+ F^+}}$$

$v^- F^-$ case:

$$\dfrac{\dfrac{}{\overrightarrow{F^-}, \triangleright \vdash \overleftarrow{F^-}}\ \text{Part 2}}{\triangleleft \twoheadrightarrow \overrightarrow{F^-}; \triangleleft\triangleright \vdash \overleftarrow{F^-}}$$

$$\dfrac{}{\overrightarrow{v^- F^-}; \triangleleft\triangleright \vdash \overleftarrow{v^- F^-}}$$

Corollary 0.32 *If* $\overleftarrow{\Gamma}; \triangleleft\triangleright \vdash \overrightarrow{A}$ *and* $\overleftarrow{\Gamma}, \overleftarrow{A}; \triangleleft\triangleright \vdash C$*, then* $\overleftarrow{\Gamma}; \triangleleft\triangleright \vdash C$.

Proof From part 4 of the above we can also derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overrightarrow{A}; \triangleleft\triangleright \vdash \overleftarrow{A}}}{\overrightarrow{A} \vdash \overleftarrow{A}}}{\triangleleft\triangleright \twoheadrightarrow \overrightarrow{A}, \triangleleft\triangleright \vdash \overleftarrow{A}}}{(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{A}) \vdash \overleftarrow{A}}}{!(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{A}) \vdash !\overleftarrow{A}} \qquad \dfrac{}{\triangleleft\triangleright \vdash \triangleleft\triangleright}$$

$$\dfrac{}{!(\triangleleft\triangleright \twoheadrightarrow \overrightarrow{A}), \triangleleft\triangleright \vdash \triangleleft\triangleright \bullet !\overleftarrow{A}}$$

By a pair of cuts in ordered logic, we obtain the desired result. ■

Ok, so there are a couple problems with ! in the 'backwards' directions of 1 and 2, which are essentially about cut. Probably a bang needs to go in $\overrightarrow{d^{\perp}}$ not in part 1.

2008.5.11

Think I fixed up the stuff from yesterday. Turns out the cut lemma needs to not quite be symmetric with identity. Bummer.

Still don't know how to properly reason about CPS-converted stuff. Should go back and look at Petri net encodings.

2008.5.13

Idea from Tom: 'play' and 'stop' terminology for focusing.

2008.5.14

Hacking on Twelf unification.

Debugging 'technique' (i.e. gross hack) for unification: Add UnifyTrail and Print to the toplevel sources.cm and also to the sources.cm inside frontend. Add to unify.sml:

```
val debug : (IntSyn.dctx * IntSyn.Exp -> string) option ref
            = ref NONE
fun trace x = print (Option.valOf (!debug) x)
```

And to unify.sig:

```
val debug : (IntSyn.dctx * IntSyn.Exp -> string) option ref
val trace : IntSyn.dctx * IntSyn.Exp -> unit
```

And then after compiling run

```
UnifyTrail.debug := SOME Print.expToString;
```

And then trace can be called from inside unify.

Note to self: trying to print the expression half of an eclo totally isn't going to work, because it refers to variables in the closure of course. Whnf it.

Oh, but Whnf returns another eclo. I suppose I should get Print to actually print the entire eclo. The constructor EClo takes it back into exp?

Excellent, turns out that works. Apparently $X1 = X2\ x$ is the unification problem coming in at the top-level and it's still failing to be solved. Is this merely because I didn't implement pruning?

To checkout twelf:

```
svn co https://cvs.concert.cs.cmu.edu/twelf/trunk twelf
```

Thinking about the thing Frank sketched:

$$\frac{\Delta;\Gamma \vdash C[p]}{\Delta;\Gamma \vdash C[\div p]}$$

$$\frac{\Delta;\Gamma, A[\alpha] \vdash C[\div\alpha]}{\Delta;\Gamma, \Diamond A[p] \vdash C[\div p]} \qquad \frac{\Delta;\Gamma \vdash C[\div p]}{\Delta;\Gamma \vdash \Diamond C[p]}$$

$$\frac{\Delta, A;\Gamma \vdash C[p]}{\Delta;\Gamma, \Box A[p] \vdash C[p]} \qquad \frac{\Delta;\Gamma \vdash C[a]}{\Delta;\Gamma \vdash \Box C[p]}$$

$$\frac{A \in \Delta \qquad \Delta;\Gamma, A[q] \vdash C[p]}{\Delta;\Gamma \vdash C[p]}$$

2008.5.15

Thinking about how to represent Pfenning-Davies even more primitively than Frank's recent encoding, by translation into a located linear logic with first-order connectives.

$$\Delta;\Gamma \vdash C[p] \qquad \longmapsto \qquad \mathbb{\Box}\Delta, \Gamma; C \to t[p] \vdash t[p]$$
$$\Delta;\Gamma \vdash C[\div p] \qquad \longmapsto \qquad \mathbb{\Box}\Delta, \Gamma; C \to t[p], w[\alpha] \vdash t[p]$$

$$\mathbb{\Box}A = \forall\alpha.(A@\alpha)$$
$$\mathbb{\Diamond}A = \exists\alpha.(A@\alpha)$$

$$\bigcirc A = (A \to \mathbb{\Box}t) \multimap t$$
$$\Diamond A = (\exists\alpha.(A \to \mathbb{\Box}t)@\alpha) \multimap w$$
$$\Box A = \bigcirc(\forall\alpha.A@\alpha)$$

$$\frac{\Gamma, A[p]; C \to t[p] \vdash t[p] \qquad \Gamma; t[p] \vdash t[p]}{\Gamma, \Diamond A[p]; C \to t[p] \vdash t[p]}$$

$$\cfrac{\cfrac{\cfrac{}{\Gamma; (\exists\alpha.(A \to \mathbb{\Diamond}w)@\alpha) \vdash \mathbb{\Diamond}w}}{\Gamma; \exists\alpha.(A \to \mathbb{\Box}t)@\alpha \vdash w[p]}}{\Gamma; \cdot \vdash \Diamond A[p]}$$

$$\frac{\Delta;\Gamma, A[\alpha] \vdash C[\div\alpha]}{\Delta;\Gamma, \Diamond A[p] \vdash C[\div p]} \qquad \frac{\Delta;\Gamma \vdash C[\div p]}{\Delta;\Gamma \vdash \Diamond C[p]}$$

Not quite working. Think I need a mixture of something like $p \otimes (p \multimap A)$ and the CPS monadic encoding.

2008.5.16

I think I got it.

$$(\Diamond A)^\star = q \multimap (\exists\alpha.(q \otimes A^\star)@\alpha)$$
$$(\Box A)^\star = p \multimap (p \otimes \forall\alpha.A^\star@\alpha)$$
$$(A \Rightarrow B)^\star = p \multimap (p \otimes (A^* \Rightarrow B^*))$$

and similar to $\Rightarrow$ for other connectives.

Does one really need to make the forall embedded in the definition of $\Box$ feel asynchronous on the left? This is what Frank does, but I don't see its necessity.

2008.5.17

Deepak suggested yesterday maybe even linearity could be dispensed with if you used enough first-order quantifiers.

2008.5.21

Something gone weird with the encoding of JS4; can't tell whether the sequent calculus should allow left decompositions 'in the monad', or at different worlds, or both, or neither. One example to watch out for besides the standard

$$\Diamond A \to \Box B \vdash \Box(A \to B)$$

is the similar

$$A \to \Box B \vdash \Box(\Box A \to B)$$

which should not be provable even in Simpson S4, but should be in S5.

2008.5.23

$$A^\dagger = t \multimap (t \otimes A^*)$$
$$(A \wedge B)^* = A^\dagger \wedge B^\dagger$$
$$(\Diamond A)^* = w \multimap (\exists\alpha.(w \otimes !A^\dagger)@\alpha)$$
$$(\Box A)^* = \forall\alpha.A^\dagger@\alpha$$

$$\Delta; \Gamma \vdash A\mathrm{t}[p] \qquad \leftrightarrow \qquad (\Box\Delta)^*, \Gamma^\dagger; t[p] \vdash (t \otimes A^*)[p]$$
$$\Delta; \Gamma \vdash A \text{ poss } [p] \qquad \leftrightarrow \qquad (\Box\Delta)^*, \Gamma^\dagger; w[p] \vdash \exists\alpha.(w \otimes A^\dagger)@\alpha[q]$$

2008.5.24

$$\frac{\Gamma; f(x) \vdash A}{\Gamma; x \vdash \$A} \qquad \frac{\Gamma, A; g(y) \vdash C}{\Gamma, \$A; y \vdash C}$$

We expect theorems

$$\frac{}{\Gamma, A; \epsilon \vdash A} \qquad \frac{\Gamma; x \vdash A \qquad \Gamma, A; y \vdash C}{\Gamma; x * y \vdash C}$$

and have an identity rule

$$\frac{}{\Gamma, a; \epsilon \vdash a}$$

To do the identity cases we want $\epsilon * x = x = x * \epsilon$.

For the connective:

Principal: $f(x) * g(y) = x * y$ (if $g(y) \downarrow$ and $f(x) \downarrow$)

RC: $f(x * y) = x * f(y)$ (if $f(y) \downarrow$)

LCL: $g(x * y) = g(x) * y$ (if $g(x) \downarrow$)

LCR: $g(x * y) = x * g(y)$ (if $g(y) \downarrow$)

2008.5.25

Truth-functional, classical interpretations of logic seem very bottom-up, or rather, the *type at which they are bottom-up* are boolean functions over free propositional variables. This is the only kind of information that gets passed up; the interface through which other connectives view them.

Is there some answer to the question of 'what gets passed up' for constructive logic that isn't the *full intension* of the proposition, or else some nasty blob of modal, semantic data?

2008.5.26

Deepak mailed me his authorization logic thing. I wonder about the difference between a system that has lots of modalities with one distinguished one that is truth full stop, and his, which seems to be much more affine about principals.

2008.5.27

The main difficulty in figuring out once and for all what connectives can be created from operations on context full of rather arbitrary judgmental gadgets is figuring out what the cut and identity principles ought to be.

2008.5.29

Let M be a manifold.

A *derivation* in M at x is a function $D : C^\infty(M) \multimap \mathbb{R}$ such that

$$D(fg) = (Df) \cdot g(x) + f(x) \cdot (Dg)$$

The tangent bundle of M is defined as the collection of derivations at each point.

Let I_x be the ideal $\{f \in C^\infty(M) \mid f(x) = 0\}$. The cotangent bundle of M is defined as $T^*_x M = I_x/I_x^2$.

We want to show that these two are actually dual.

Given a derivation D, we generate a dual element of the cotangent bundle (a map $I_x/I_x^2 \multimap \mathbb{R}$) by simply taking in an element of I_x and hitting it with D. Why does this respect the quotient by I_x^2? Because by the definition of product ideal any $f \in I_x^2$ decomposes as $\sum_i f_i g_i$ for $f_i, g_i \in I_x$, and so $D(f) = D(\sum_i f_i g_i) = \sum_i f_i(x) Df_i + g_i(x) Dg_i = 0$.

Given a map $r : I_x/I_x^2 \multimap \mathbb{R}$, we can generate a derivation by setting $D(f) = r(f - f(x) + I_x^2)$. This is pretty obviously linear, and we are certainly handing r something that's zero at x, so let's check the Leibniz law.

$$\begin{aligned} D(fg) &= r(gf - g(x)f(x) + I_x^2) \\ &= r(f(x)g + g(x)f - 2g(x)f(x) + I_x^2) \\ &= r(f(x)g - f(x)g(x) + I_x^2) + r(g(x)f - g(x)f(x) + I_x^2) \\ &= r(g - g(x) + I_x^2)f(x) + r(f - f(x) + I_x^2)g(x) \\ &= (Dg)f(x) + (Df)g(x) \end{aligned}$$

With the first equality being justified by

$$(g - g(x))(f - f(x)) = gf - f(x)g - g(x)f + g(x)f(x) \in I_x^2$$

Finally we want to believe these two transformations are inverse to one another. Starting with a derivation D, turning it into a co-cotangent vector $D : I_x/I_x^2 \to \mathbb{R}$, and turning it back into a derivation, we get $D(f - f(x) + I_x^2) = Df - 0 + 0$. (Why is a derivation is zero on constant functions? Leibniz says $D1 = D(1 \cdot 1) = 1D1 + 1D1 = 2D1$, so $D1 = 0$, and by linearity all constant functions must yield zero).

Starting with a co-cotangent $r : I_x/I_x^2 \to \mathbb{R}$, we turn it into a derivation $r(f - f(x) + I_x^2)$ and then back into a co-cotangent by restricting f to be in I_x/I_x^2. Then $r(f) = r(f - f(x) + I_x^2) = r(f + I_x^2)$.

2008.5.30

Remember that the cause of my bug might be composition of substitutions. There is no guarantee of η-longness internally.

2008.6.1

Lagrangian mechanics take place in the tangent bundle TM of position space; Hamiltonian in the cotangent bundle T^*M.

I think of tangent vectors as being sort of more 'dual' than 'natural' since they are derivations on scalar functions. But the established terminology runs opposite to that; covectors are dual to vectors, and they are the things that differentials must be.

2008.6.2

I have a pile of loose research ends, and at any time I do not know what they are. Doing a bit of bookkeeping helped me *see* clearly where my money was going, allowed me to see for instance how much I was actually devoting to food. I should like to do the same sort of flashlight-waving at various other mental habits/structures.

It seems to remain important to me that I feel like I *own* a project to feel inclined to work on it. Ownership is not necessarily exclusive; I am intrinsically motivated to work on this telegraph key hack with lea, because it is fun. Nor the converse: my thesis project is highly self-directed, but the obligation to do it feels external.

I really would like a nice mechanical proof of correctness of certain linear token-passing protocols I've come up with — I ought to lean on these for motivation.

Loose ends that I can think of right now:

- diagrammatic notation for sequents
- simple-type polymorphism in LF
- contextual form in LF
- token-translation completeness of focusing

related to thesis work

- unification implementation
- labelled unification
- monotonicity

It's difficult to publish something like simple-type polymorphism that has unclear application, and many known disadvantages, but I should like to *publicize* it all the same.

2008.6.5

Interesting stuff about units in 'Elements of Physics' by Tarantola. He seems to claim at achieve agnosticism not only on the issue of, say, m vs. cm, but also m vs. m^2.

2008.6.6

It seems possible that proof irrelevance as a logic is just two independent monads composed together. Proving soundness of this translation is strangely tricky, though.

2008.6.7

To show: that a proof in

$$\frac{\Gamma \vdash A\ m_i}{\Gamma \vdash \bigcirc_i A} \qquad \frac{\Gamma, A \vdash C\ m_i}{\Gamma, \bigcirc_i A \vdash C\ m_i} \qquad \frac{\Gamma \vdash C}{\Gamma \vdash C\ m_i}$$

implies a proof in

$$\frac{\Gamma \vdash A^{\div}}{\Gamma \vdash \star A} \qquad \frac{\Gamma, A \vdash C^{\div}}{\Gamma, A^{\div} \vdash C^{\div}} \qquad \frac{\Gamma, A^{\div} \vdash C}{\Gamma, \star A \vdash C} \qquad \frac{\Gamma \vdash C}{\Gamma \vdash C^{\div}}$$

2008.6.8

Is there any way to think of the quantity flows in bookkeeping as flux across a 'spacetime' region, considered very metaphorically?

2008.6.9

Consider nondeterministic translation of the bimonadic system into proof irrelevance.

$$\frac{A \mapsto^{-} B}{\bigcirc_1 \bigcirc_2 A \mapsto^{-} \star B} \qquad \frac{A \mapsto^{+} B}{\bigcirc_1 \bigcirc_2 A \mapsto^{+} \star B}$$

$$\frac{A \mapsto B}{\bigcirc_2 A \mapsto^{-} B^{\div}} \qquad \frac{A \mapsto B}{\bigcirc_2 A\ m_1 \mapsto^{+} \star B}$$

$$\frac{A \mapsto B}{\bigcirc_2 A \mapsto^{+} \star B}$$

$$\frac{A \mapsto B}{A\ m_2 \mapsto^{+} B^{\div}}$$

$$\frac{A \mapsto^+ B}{\bigcirc_2 A\ m_1 \mapsto^+ B^{\div}} \qquad \frac{A \mapsto^- B}{\bigcirc_2 A \mapsto^- B}$$

Lemma 0.33 *If* $(\Gamma \vdash A) \mapsto (\Gamma' \vdash A')$ *and* $\Gamma \vdash A$ *then* $\Gamma' \vdash A'$.

2008.6.10

It's frustrating how this proof doesn't seem to yield a good focussing story, still; for proof-irrelevance as a modality is plainly a connective, but two monads in a row are not unipolar.

2008.6.11

Seems to be easier to treat all of proof-irrelevant logic as a 'monad-translation' image of a logic with *one* monad, kind of like how classical logic is double-negation-translated constructive logic.

2008.6.12

Some aspects of type reconstruction for HLF seem like I worked them out long ago, but I have a hard time concretely picturing the algorithm. How can I get an implementation up and running quickly so that I can see whether, say, the proof of cut elimination can be effectively reconstructed or not?

I worry a little bit about ampersand and top, which I have not thought carefully about in many places.

2008.6.13

$$\frac{\dfrac{\dfrac{\Gamma, NA \vdash C}{\Gamma, NA \vdash C\ \dot{m}}}{\Gamma, \odot NA \vdash C\ \dot{m}}}{}$$

$$\frac{\Gamma \vdash A \qquad \dfrac{\Gamma, A \vdash C^{\div}}{\Gamma, A \vdash \triangle C}}{\Gamma \vdash \triangle C}$$

$$\frac{\dfrac{\Gamma, B^{\div} \vdash A}{\Gamma, \triangle B \vdash A} \qquad \Gamma, A \vdash C^{\div}}{\Gamma \vdash C^{\div}}$$

More like:

$$\frac{\Gamma, A^?, B^? \vdash \gamma}{\Gamma, A \wedge B \vdash \gamma} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A^? \vdash C} \qquad \frac{\Gamma \vdash C^{\div}}{\Gamma \vdash \triangle C} \qquad \frac{\Gamma, A^? \vdash C^{\div}}{\Gamma, \triangle A \vdash C^{\div}} \qquad \frac{\Gamma \vdash C}{\Gamma \vdash C^{\div}}$$

2008.6.14

Focussed syntax of indexed monadic logic:

$$\begin{array}{llll} \text{Positive} & A^+ & ::= & \downarrow A^- \mid A^+ \otimes B^+ \\ \text{Negative} & A^- & ::= & \uparrow A^+ \mid A^+ \multimap B^- \mid \bigcirc_i^- E_i \\ \text{Monadic} & E_i & ::= & \bigcirc_i^+ A^+ \end{array}$$

The translation I expect wants to wrap the truth-monad around positive things — at least, in that boundary where negative things are outside and positive things are inside. The steady state is

$$\Gamma^- \vdash \odot^+ A^+ \ \dot{m}$$

If we focus on something on the left, we'll eventually hit an up-shift, which is where the translation ought to insert a $\odot^-\odot^+$. This will appropriately require $\dot{m}$ on the right. After that will be asynchronous work. If we focus on the right, we'll continue through the positive stuff, and then asynchronously tear through negative work, finally decomposing $\odot^-$.

Question: is it the negative finish or the positive begin that translates to $\odot^-\odot^+$? Or is it one each?

2008.6.15

I think if I replace $\uparrow$ by $\bigcirc$ and $\downarrow$ by $\Box$, I preserve provability and restrict proofs in a way that sort of approximates focussing.

Consider logic with the connectives partially polarized

$$\begin{array}{llll} \text{Positive} & A^+ & ::= & A^- \mid A^- \otimes B^- \mid A^- \oplus B^- \mid 1 \mid 0 \mid p^+ \\ \text{Negative} & A^- & ::= & \uparrow A^+ \mid A^+ \multimap B^- \mid A^- \mathbin{\&} B^- \mid \top \mid p^- \end{array}$$

but with just standard unfocussed rules. Notice that positive formulae here are strictly more general than negative, and that usually positive connectives take negative arguments because they do not effectively chain. We have for the shift simply

$$\frac{\Gamma \vdash A^+}{\Gamma \vdash \uparrow A^+} \qquad \frac{\Gamma, A^+ \vdash C^+}{\Gamma, \uparrow A^+ \vdash C^+}$$

So consider replacing that shift with $\bigcirc$:

$$\frac{\Gamma \vdash A^+}{\Gamma \vdash A^+ \ m} \qquad \frac{\Gamma \vdash A^+ \ m}{\Gamma \vdash \bigcirc A^+} \qquad \frac{\Gamma, A^+ \vdash C^+ \ m}{\Gamma, \bigcirc A^+ \vdash C^+ \ m}$$

Plainly we prove no *more* things in this system. For if there is a proof in that system, just erase all monadic judgments to truth. We may also see that we prove no *fewer*, as long as the correspondence is with sequents $A^+, \cdots, A^+ \vdash A^-$ in the original calculus. For if there is a proof in the original calculus, there is one that eagerly does all asynchronous decompositions, including tearing off $\uparrow$ on the right.

If $\Gamma \vdash A^-$, then $\Gamma^* \vdash (A^-)^*$.

If $\Gamma \vdash A^+$, then $\Gamma^* \vdash (A^+)^*\ m$.

Like in twelf:

```
pos : type.
neg : type.

inj : neg -> pos.
up : pos -> neg.
tensor : neg -> neg -> pos.
oplus : neg -> neg -> pos.
amp : neg -> neg -> neg.
lol : pos -> neg -> neg.
top : neg.
1 : pos.
0 : pos.

t : type.
shift : t.
circ : t.

phyp : pos -> type.
nhyp : neg -> type.
pconc : t -> pos -> type.
nconc : t -> neg -> type.
mon : pos -> type.

tensorL : (phyp (tensor A B) -> pconc T C)
   <- (nhyp A -> nhyp B -> pconc T C).
tensorR : pconc T (tensor A B)
  <- nconc T A
  <- nconc T B.
injL : (phyp (inj A) -> pconc T C) <- (nhyp A -> pconc T C).
injR : pconc T (inj A) <- nconc T A.
shiftL : (nhyp (up A) -> pconc shift C)
           <- (phyp A -> pconc shift C).
```

```
shiftR : nconc shift (up A) <- pconc shift A.
circL : (nhyp (up A) -> mon C) <- (phyp A -> mon C).
circR : nconc T (up A) <- mon A.
monR : mon A <- pconc circ A.
```

2008.6.16

Quick little perl script that iteratively chunks bigrams that are way more frequently occurring than would be expected from the relative frequencies of their components.

```
#!/usr/bin/perl
use strict;

sub learn {
    my (@t) = @_;
    my (%freq, %bigram, %rat, %good);

    $freq{$_}++ for @t;
    for (1..$#t) {
$bigram{$t[$_-1] . ";" . $t[$_]}++;
    }
    for (sort {$bigram{$b} <=> $bigram{$a}} keys %bigram) {
my ($x, $y) = split /;/, $_;
my $act = $bigram{$_} / (@t - 1);
my $exp = ($freq{$x} / @t) * ($freq{$y} / @t);
$rat{$_} = $act / $exp;
    }

#    for (sort {$freq{$b} <=> $freq{$a}} keys %freq) {
# print "$_: $freq{$_}\n";
#    }

    for (sort {$rat{$b} <=> $rat{$a}} keys %rat) {
my $p = $_;
$p =~ s/;//;
if ($bigram{$_} >= 5 && $rat{$_} >= 5.0) {
    $good{$_} = 1;
    print "$p: $bigram{$_} $rat{$_}\n";
}
    }

    my @out;
```

```
    while (@t) {
my $head = shift @t;
if (@t) {
    my $cand = "$head;$t[0]";
    if ($good{$cand}) {
shift @t;
$cand =~ s/;//;
push @out, $cand;
    }
    else {
push @out, $head;
    }
}
else {
    push @out, $head;
}
    }
    return @out;
}

my $txt;

{local $/; $txt = <>}

$txt =~ s/ /  /g;
my @c = split //, $txt;

for (1..18) {
    @c = learn(@c);
    print "--\n";
}
print join "", map {length() > 1 ? "($_)" : $_} @c;
```

2008.6.17

I expect the proof-irrelevant modality to wind up as $\odot\bigcirc$, but it's still a mystery to me why that system works at all when I can't come up with a 'synthetic' system that satisfies cut.

It seems to require synchronous focussing reasoning on the left to prove it correct — specifically to see that negative connectives can't be left-decomposed during the monadic promotion phase.

2008.6.18

Have been reading 'The Design of Everyday Things'. I am a little skeptical of the way Donald Norman uses the word 'logic'. He seems to use it to denote those inferences that people 'naturally' make when trying to figure out the mapping between controls and behavior. What's clear is that if you toss a device in front of a sample of people, they'll do *something*, but it seems hard to me to separate what is culturally learned from what is intrinsically 'natural' or 'logical' or whatever.

Another nearby concept is *iconicity*, the peculiar property of the isomorphism between well-laid-out controls on a stovetop and the burners: a spatial isomorphism that only involves scaling and translation, and no rotation. I find it hilariously ironic that on pages 76-77 of DOET, the very diagrams that are criticizing label-dependent arbitrary layouts require *labels* to connect them to the pieces of text on the same page that describe them, for they are not laid out in an isomorphic way.

I think the lesson is, anyway: know your audience, and maximize the chance that they will guess the right thing. *Study* your intended audience to find out the patterns of their guesses. There *may* be reliable rules of thumb as to what they will guess, but introspection is, as a rule, a terrible way of finding them, and constant empirical reevaluation seems quite important.

I expect to translate

$$\begin{array}{rcl} \text{Props } A & ::= & A \Rightarrow \triangleleft E \mid \triangleleft E \\ \text{Monadic Props } E & ::= & \triangleright(A \oplus B) \mid \triangleright(A \otimes B) \mid \triangleright A \end{array}$$

$$\frac{\Gamma, A \vdash E\ m}{\Gamma \vdash A \Rightarrow \triangleleft E} \qquad \frac{\Gamma \vdash A \qquad \Gamma, E\ m \vdash F\ m}{\Gamma, A \Rightarrow \triangleleft E \vdash F\ m}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash \triangleright(A \otimes B)\ m} \qquad \frac{\Gamma, A, B \vdash F\ m}{\Gamma, \triangleright(A \otimes B)\ m \vdash F\ m}$$

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash \triangleright(A_1 \oplus A_2)\ m} \qquad \frac{\Gamma, A_1 \vdash F\ m \qquad \Gamma, A_2 \vdash F\ m}{\Gamma, \triangleright(A_1 \oplus A_2)\ m \vdash F\ m}$$

Can I prove identity for focussing without tokens, as long as I have 'standard' connectives floating around?

$$\begin{array}{rcl} \text{Props } A & ::= & \cdots \mid A \otimes B \mid v^+ F^+ \\ \text{Positive Props } F^+ & ::= & \cdots \mid F^+ \otimes F^+ \mid d^+ A \end{array}$$

Want to show $A \vdash A$ for all A.

So show $\underline{v^+ F^+} \vdash v^+ F^+$ for all F^+. So I want to be able to cut in something like '$(F^+)^* \vdash \underline{v^+ F^+}$' where $*$ turns a focused proposition back to normal.

2008.6.19

Let there be a preorder $(P, \leq)$, typical element x.

Syntax is

$$\text{Props } A \quad ::= \quad F_x A \mid U_x A \mid A \wedge B \mid A \vee B \mid A \Rightarrow B$$

The judgment is $\Gamma \vdash A[x]$ where Γ is composed of $B[y]$ where by invariant each $y \geq x$.

$$\frac{A \text{ prop}[x] \qquad B \text{ prop}[x]}{A \star B \text{ prop}[x]} \qquad \frac{y \leq x \qquad A \text{ prop}[x]}{F_x A \text{ prop}[y]} \qquad \frac{y \leq x \qquad A \text{ prop}[y]}{U_y A \text{ prop}[x]}$$

$$\frac{\Gamma\big|_{\geq x} \vdash A[x]}{\Gamma \vdash F_x A[y]} \qquad \frac{\Gamma, A[x] \vdash C[z]}{\Gamma, F_x A[y] \vdash C[z]}$$

$$\frac{\Gamma \vdash A[y]}{\Gamma \vdash U_y A[x]} \qquad \frac{y \geq z \qquad \Gamma, A[y] \vdash C[z]}{\Gamma, U_y A[x] \vdash C[z]}$$

$$\frac{\Gamma, A[x] \vdash B[x]}{\Gamma \vdash A \Rightarrow B[x]} \qquad \frac{\Gamma \vdash A[x] \qquad \Gamma, B[x] \vdash C[z]}{\Gamma, A \Rightarrow B[x] \vdash C[z]}$$

$$\bigcirc_{xy} A = U_y F_x A \qquad \frac{y \leq x \qquad A \text{ prop}[x]}{\bigcirc_{xy} A \text{ prop}[x]}$$

$$\Box_{xy} A = F_x U_y A \qquad \frac{y \leq x \qquad A \text{ prop}[y]}{\Box_{xy} A \text{ prop}[y]}$$

$$\Diamond_{xy} A = \Box_{xy}(A \multimap p) \multimap p$$

$$\triangle_{xy} A = F_x U_y F_x A \qquad \frac{y, z \leq x \qquad A \text{ prop}[x]}{\triangle_{xy} A \text{ prop}[z]}$$

2008.6.20

The thing from yesterday is *so* close to Deepak's system, but not quite the same. Not sure what is going on.

2008.6.21

Consider the following two sequences:

(A) The number of downward closed subsets of the powerset of n. (2, 3, 6, 20, 168, 7581, 7828354)

(B) The number of antichain covers of n. That is, set covers of n that include no pair of subsets that are related to each other by $\subseteq$. (1, 2, 9, 114, 6894, 7785062)

(B) is also the number downward closed subsets of the powerset of n that include all singletons. More trivially, (A) is also the number of monotone boolean functions of n variables. These are apparently called the 'Dedekind numbers' and there's no known nice closed form for them.

Turns out (A) is the inverse binomial transform of what you get if you stick a 2 in front of (B). That is, the 'derivative of (A) at time 1' is 3-2=1. The 'second derivative' is (6-3)-(3-2) is 2. The 'third derivative' is ((20-6)-(6-3))-((6-3)-(3-2))=9. etc.

2	1	2	9	114	6894
3	3	11	123	7008	
6	14	134	7131		
20	148	7265			
168	7413				
7581					

If we take only nonempty downward closed subsets

1	1	2	9	114	6894
2	3	11	123	7008	
5	14	134	7131		
19	148	7265			
167	7413				
7580					

This makes more sense as there ought to be exactly one antichain cover of $\emptyset$ as well as $\{*\}$.

Oh this whole tableau makes sense in a rather simple way. Let c_{nk} be the numberof antichains not containing the empty set in a set of $n+k$ labelled elements, where the first k of them must be covered. Then $c_{(n+1)k} = c_{n(k+1)} + c_{nk}$ — consider one of the n elements that might be covered or not. Either it is covered and we consider an antichain with one more covered and one fewer maybe-covered ($c_{n(k+1)}$) or else it is not covered and we need to come up with an antichain with one fewer maybe-covered (c_{nk}). c_{n0} is (A)-1 and c_{0k} is (B).

2008.6.22

Schur's lemma says any map $\phi : \mathbb{C}^n \multimap \mathbb{C}^n$ that commutes with an irreducible finite-dimensional representation $\rho : G \to (\mathbb{C}^n \multimap \mathbb{C}^n)$ of a group G is a scalar.

Why? Well, the fundamental theorem of algebra says ϕ has an eigenvalue, call it λ. How big is the eigenspace E_λ? Well, it's left invariant by the action of ρ. For let $g \in G$ be given, and $v \in E_\lambda$. We want to confirm that $\rho g v \in E_\lambda$. So notice $\phi(\rho g v) = \rho g(\phi v) = \rho g(\lambda v) = \lambda(\rho g v)$. Since ρ was assumed irreducible, E_λ can't be any smaller than the whole space: $E_\lambda = \mathbb{C}^n$. So the action of ϕ on any vector is to multiply it by λ, QED.

2008.6.23

Torben Braüner points out that if we have nominals that are also propositions that are true when we arrive at that world, general downarrow is definable.

$$\downarrow x.A \equiv \forall x.(\mathbf{Here}_x \to A)$$

2008.6.24

Can the synchronous choice required by $\&, \oplus$ be decomposed as a combination of adjoint passage from one judgment to another and otherwise simple connectives? $A \,\&\, B = F(UA \otimes UB)$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A\mathsf{t} \vdash C\mathsf{t}^+}}{\Gamma, UA *, UB * \vdash C\mathsf{t}^+}}{\Gamma, UA \otimes UB * \vdash C\mathsf{t}^+}}{\Gamma, F(UA \otimes UB)\mathsf{t}^- \vdash C\mathsf{t}^+}$$

It seems inevitable that any compound 'connective' that uses any amount of transport can't uniformly be positive or negative because it must make some round-trip back to truth.

How about: keep right-hand side typically at m and

$$A \,\&\, B = F(\neg U \neg A \otimes \neg U \neg B)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, A \vdash \# \; m}{\Gamma, A \vdash U\bot}}{\Gamma \vdash \neg A}}{\Gamma, \neg F \neg B \; m \vdash F \neg A \; m}}{\Gamma, \neg F \neg A \; m, \neg F \neg B \; m \vdash \# \; m}}{\Gamma, U(\neg F \neg A \otimes \neg F \neg B)\mathsf{t} \vdash \# \; m}$$

2008.6.25

Synthetic hypersequent rule from $A \vee \neg A$: First we get

$$\frac{}{\vdash A | A \vdash}$$

then

$$\frac{G|\Gamma, \Gamma' \vdash \Pi}{G|\Gamma \vdash \Pi | \Gamma' \vdash}$$

Can we classically prove $(A \wedge B \Rightarrow C) \Rightarrow (A \Rightarrow C) \vee (\neg B)$? Yes, of course. But the converse of this rule is already available, just by local weakening and hypercontraction.

So really a multiple conclusion sequent $A_1, \ldots A_n \vdash B_1, \ldots, B_m$ is an intuitionistic hypersequent canonically

$$A_1 \vdash | \cdots | A_n \vdash | \vdash B_1 | \cdots | B_n$$

2008.6.26

Gröbner bases are sets of *multivariate* polynomials for which division still yields a unique answer. That is, we have polynomials g and $f_1, \ldots, f_n$ and we wish to divide g by the fs. We nondeterministically pick an f_i and muliply it by a polynomial enough for its leading term to cancel some term of g — this requires an ordering on monomials. Having cancelled a term of g, we will only leave behind junk that's smaller in the monomial ordering. For general bases $f_1, \ldots, f_n$ this division algorithm gives different answers, but the surprising fact is that we can pass to a Gröbner basis which generates the same ideal.

2008.6.27

Atomic propositions are just top and bottom at arbitrarily removed judgments, maybe?

Consider a positive atom. It should be able to asynchronously remain on the left. So make up a new judgment that is stronger than all the others. Say $p^+ = F^+ p$. On the left it transitions to 'p valid'?

2008.6.28

At least in linear logic, F and U satisfy the following:

$$U(FA \multimap B) = A \Rightarrow UB$$

$$U(A \mathbin{\&} B) = UA \wedge UB$$

$$FA \otimes FB = F(A \wedge B)$$

$$FA \oplus FB = F(A \vee B)$$

$$U\top = \mathrm{t}$$

$$1 = F\mathrm{t}$$

$$0 = F\mathrm{f}$$

2008.6.29

And so this 'explains' why $!(A \;\&\; B) = !A \otimes !B$, for $FU(A \;\&\; B) = F(UA \wedge UB) = FUA \otimes FUB$.

2008.6.30

It makes sense that $(A \multimap p) \multimap p$ and $p \multimap (A \otimes p)$ are both monads, because $\Box \multimap p$ is self-adjoint considered as a contravariant functor, and obviously $p \multimap \Box$ is right adjoint to $\Box \otimes p$. Notice that both propositional functions are negative on the outside, and positive on the inside.

2008.7.1

It doesn't seem quite as clear to me as it used to that polarization-as-modalization should work *both* ways around implication.

If positive is mapped to strong and negative to weak, I need to map $A^+ \Rightarrow B^-$ to something like

$$FA^{+*} \Rightarrow^w B^{-*}$$

in which case the F is contiguous with the positive argument. If, however, the mapping goes the other way around, I seem to be forced to map $B^+ \Rightarrow A^-$ to either

$$UB^{+*} \Rightarrow^s A^{-*}$$

and interrupt the first argument of $\Rightarrow$ or

$$U(B^{+*} \Rightarrow^w FA^{-*})$$

and interrupt the second.

2008.7.2

$$\begin{array}{rcl} \text{Negative Props } N & ::= & \uparrow P \mid P \Rightarrow N \\ \text{Positive Props } P & ::= & \downarrow N \mid P \vee P \mid \triangle P \\ \text{Strong Props } A & ::= & UB \mid U'B' \mid A \Rightarrow A \\ \text{Weak Props } B & ::= & FA \mid B \vee B \\ \text{Weak' Props } B' & ::= & F'A \end{array}$$

$\square^*$ maps P to B and N to A.

$$\begin{cases} P^- = UP^* \\ N^- = N^* \\ (P \div)^- = U'F'UP^* \end{cases} \qquad \begin{cases} P^+ = P^* \\ N^+ = FN^* \\ (P \div)^+ = F'UP^* \end{cases}$$

$$(\uparrow P)^* = UP^*$$
$$(\downarrow N)^* = FN^*$$
$$(P \Rightarrow N)^* = UP^* \Rightarrow UFN^*$$
$$(P_1 \vee P_2)^* = FUP_1^* \vee FUP_2^*$$
$$(\triangle P)^* = FU'F'UP^*$$

Lemma 0.34

$$\Gamma \vdash X \Leftrightarrow \Gamma^- \vdash X^+$$

Proof Right to left is easy; insertion of Us and Fs naturally only restricts proofs. Left to right involves the following cases.

$$\frac{\Gamma \vdash P}{\Gamma \vdash \uparrow P} \Leftrightarrow \frac{\dfrac{\Gamma^- \vdash P^*}{\Gamma^- \vdash UP^*}}{\Gamma^- \vdash FUP^*} \qquad \frac{\Gamma, P \vdash X}{\Gamma, \uparrow P \vdash X} \Leftrightarrow \frac{\Gamma^-, UP^* \vdash X^+}{\Gamma^-, UP^* \vdash X^+}$$

$$\frac{\Gamma \vdash N}{\Gamma \vdash \downarrow N} \Leftrightarrow \frac{\Gamma^- \vdash FN^*}{\Gamma^- \vdash FN^*} \qquad \frac{\Gamma, N \vdash X}{\Gamma, \downarrow N \vdash X} \Leftrightarrow \frac{\dfrac{\Gamma^-, N^* \vdash X^+}{\Gamma^-, FN^* \vdash X^+}}{\Gamma^-, UFN^* \vdash X^+}$$

$$\frac{\Gamma, P \vdash N}{\Gamma \vdash P \Rightarrow N} \Leftrightarrow \frac{\dfrac{\dfrac{\Gamma^-, UP^* \vdash FN^*}{\Gamma^-, UP^* \vdash UFN^*}}{\Gamma^- \vdash UP^* \Rightarrow UFN^*}}{\Gamma^- \vdash F(UP^* \Rightarrow UFN^*)}$$

$$\frac{\Gamma \vdash P \qquad \Gamma, N \vdash X}{\Gamma, P \Rightarrow N \vdash X} \Leftrightarrow \frac{\Gamma^- \vdash P^* \qquad \dfrac{\dfrac{\Gamma^-, N^* \vdash X^+}{\Gamma^-, FN^* \vdash X^+}}{\Gamma^-, UFN^* \vdash X^+}}{\Gamma^-, UP^* \Rightarrow UFN^* \vdash X^+}$$

$$\frac{\Gamma, P_i}{\Gamma \vdash P_1 \vee P_2} \Leftrightarrow \frac{\dfrac{\dfrac{\Gamma^- \vdash P_i^*}{\Gamma^- \vdash UP_i^*}}{\Gamma^- \vdash FUP_i^*}}{\Gamma^- \vdash FUP_1^* \vee FUP_2^*}$$

$$\frac{\Gamma, P_1 \vdash X \qquad \Gamma, P_2 \vdash X}{\Gamma, P_1 \vee P_2 \vdash X} \Leftrightarrow \frac{\dfrac{\Gamma^-, UP_1^* \vdash X^+}{\Gamma^-, FUP_1^* \vdash X^+} \qquad \dfrac{\Gamma^-, UP_2^* \vdash X^+}{\Gamma^-, FUP_2^* \vdash X^+}}{\dfrac{\Gamma^-, FUP_1^* \vee FUP_2^* \vdash X^+}{\Gamma^-, U(FUP_1^* \vee FUP_2^*) \vdash X^+}}$$

$$\frac{\Gamma \vdash P^{\div}}{\Gamma \vdash \triangle P} \Leftrightarrow \frac{\dfrac{\Gamma^- \vdash F'UP^*}{\Gamma^- \vdash U'F'UP^*}}{\Gamma^- \vdash FU'F'UP^*} \qquad \frac{\Gamma, P^{\div} \vdash X}{\Gamma, \triangle P \vdash X} \Leftrightarrow \frac{\dfrac{\Gamma^-, U'F'UP^* \vdash X^+}{\Gamma^-, FU'F'UP^* \vdash X^+}}{\Gamma^-, UFU'F'UP^* \vdash X^+}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P^{\div}} \Leftrightarrow \frac{\dfrac{\Gamma^- \vdash P^*}{\Gamma^- \vdash UP^*}}{\Gamma^- \vdash F'UP^*} \qquad \frac{\Gamma, P \vdash Q^{\div}}{\Gamma, P^{\div} \vdash Q^{\div}} \Leftrightarrow \frac{\dfrac{\Gamma^-, UP^* \vdash F'UQ^*}{\Gamma^-, F'UP^* \vdash F'UQ^*}}{\Gamma^-, U'F'UP^* \vdash F'UQ^*}$$

I seem to be putting in a lot of interruptions into the translations of connectives in order to make the proof easy even without focussing reasoning. Perhaps this focussing reasoning can be expressed merely as identities about the adjoint connectives, combined with idempotency of the monad?

Do I get for instance

$$FU(FUB_1 \vee FUB_2) = FUB_1 \vee FUB_2$$

?

$$\frac{\dfrac{FUB_1 \vee FUB_2 \vdash FUB_1 \vee FUB_2}{U(FUB_1 \vee FUB_2) \vdash FUB_1 \vee FUB_2}}{FU(FUB_1 \vee FUB_2) \vdash FUB_1 \vee FUB_2}$$

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{UB_i \vdash UB_i}}{UB_i \vdash FUB_i}}{UB_i \vdash FUB_1 \vee FUB_2}}{UB_i \vdash U(FUB_1 \vee FUB_2)}}{UB_i \vdash FU(FUB_1 \vee FUB_2)}}{FUB_i \vdash FU(FUB_1 \vee FUB_2)} \quad \forall i}{FUB_1 \vee FUB_2 \vdash FU(FUB_1 \vee FUB_2)}$$

Yes, apparently! In fact,

$$FU(FA_1 \vee FA_2) = FA_1 \vee FA_2$$

(Back in Pfenning-Davies land, we'd just say $\Box(\Box A \vee \Box B) = \Box A \vee \Box B$) which I can explain by

$$FU(FA_1 \vee FA_2) = FUF(A_1 \vee A_2) = F(A_1 \vee A_2) = FA_1 \vee FA_2$$

Because generally

$$FUF = F \qquad UFU = U$$

So it seems I can characterize positive connectives as those which are impervious to FU and negative connectives as those that are impervious to UF. This *seems* to allow the simplified clauses

$$(P \Rightarrow N)^* = UP^* \Rightarrow N^*$$

$$(P_1 \vee P_2)^* = P_1^* \vee P_2^*$$

but as soon as I make that definition, they lose that property, don't they? Or maybe I'm not using the induction hypothesis properly. *Supposing* that $FUP_i = P_i$, then $FU(P_1 \wedge P_2) = FU(FUP_1 \wedge FUP_2) = FUF(UP_1 \wedge UP_2) = F(UP_1 \wedge UP_2) = FUP_1 \wedge FUP_2 = P_1 \wedge P_2$. Okay, perfect.

Being positive or negative on the *inside* can be characterized by invariance under replacing an argument X with FUX or UFX, respectively?

Clearly connectives 'imported' from the other strength trivially satisfy this property (e.g. negative $\uparrow P$ is translated to UP^*, so it's trivially negative because $UFUP^* = UP^*$) so it is a generalization to propositions that 'effectively' are imported.

Oh, this is just the trivial fact that: $FUB = B$ just in case there *exists* an A such that $FA = B$. In one direction, UB is the witness, otherwise, $FUB = FUFA = FA = B$.

Still seems like it ought to be possible to modally capture more of focussing discipline, especially somehow incorporating netural propositions.

Consider a preorder with neutral in the middle and two judgments above and below it. Could one map a negative phase, say, to first jumping up and spending most of your time below, and a positive phase to jumping down and spending most of your time above? Or vice-versa?

The troubling thing about the preorder discipline is that it requires one to stabilize on high things on the left and low things on the right — bad conditions for trying to stabilize on neutrality on both.

The other goal with this is to show that I can push all the positive connectives (and the argument of implies) back into the strong judgment to show that the proof irrelevant modality is the two-monad sequence without translating anything else.

I found what I think is a Celf bug:

```
o : type.
k : o.
eq : o -> o -> type.
refl : eq A A.
down : (((o -> o) -> o) -> o) -> o.
q : type.
rule : q <- eq (down V) (down (\q:((o -> o) -> o).
                 q (\r:o.k)))
         <- eq U (V (\y:o -> o.y U)).

#query 10 * * 1 q.
```

finds a solution, but with the simple reversal of goal order

```
rule : q <- eq U (V (\y:o -> o.y U))
         <- eq (down V) (down (\q:((o -> o) -> o).
                q (\r:o.k))).
```

it does not. I strongly suspect the problem is Anders is taking the equations

$$u[] = v[\lambda y.y\ u[]]$$

$$v[q] = q\ (\lambda r.k)$$

and transforming the first, via overzealous lifting of _, to

$$u[] = v[\_]$$

thereby invalidating the second.

2008.7.3

If unification is just left rules for equality, then it is a funny kind of binary judgment, which becomes effectively bottom upon disequality. I don't suppose there's any way of connecting the substructural-nominal intuition to this? It's difficult because it also depends on positive evidence of equality as well as disequality.

2008.7.4

I feel like I can spin disjunction as taking negative arguments and translate it

$$(N_1 \vee N_2)^* = FN_1^* \vee FN_2^*$$

2008.7.5

This seems more difficult with implication. Basically sure that I want positive on the right to achieve the same effect, but what about on the left? Possibly also positive?

$$(P_1 \Rightarrow P_2)^* = UP_1^* \Rightarrow UP_2^*$$

On the right (beginning with $F(UP_1^* \Rightarrow UP_2^*)$) we synchronously bite off the F and then take the asynchronous decompositions all the way to the appropriate $UP_1^* \vdash P_2^*$, and on the left, we get the two goals $\vdash UP_1^*$ and (assuming that we *don't* focus through the U) also $UP_2^* \vdash C$. The first goes to just $\vdash P_1^*$ asynchronously.

2008.7.6

So yeah! I don't actually mean to sequence the $\vee$ and the Fs together on the right to accurately simulate unfocussed reasoning.

2008.7.7

Aleksey emailed me about being interested in proof irrelevance encoded as two monads. Don't quite understand what he's talking about yet.

2008.7.8

At least now I sort of understand why he wants to weaken the fixpoint rule — it seems strikingly similar to both callcc and Löb logic somehow. I can't seem to hit it with the Ciabattoni hammer, since it appears cyclic.

2008.7.9

I refigured out the translation from a week ago from scratch. The notion that one direction of the proof comes for free is basically wrong, but everything still works out — they key is to synchronously combine the $\Rightarrow$ and inner right U in decomposing $UP^* \Rightarrow UFN^*$ on the left, for instance.

2008.7.10

Given the freedom I have to move Us around negative things and Fs around positive things, it seems I can put the monad UF around positive things and cast the proof irrelevance monad as either $UFU'F'$ or $U'F'UF$. Why can or can't I find a counterexample to the possibility of erasing most UFs?

No, that's not true — the more I think about it, I am pretty convinced that if proof-irrelevance is to be two monads, the remaining stuff must be scattered UFs, and triangle comes out as $U'F'UF$, not as $UFU'F'$.

I'm struck suddenly by the symmetry in

$$\frac{\Gamma \vdash A^{\div}}{\Gamma \vdash \triangle A} \qquad \frac{\Gamma, A^{\div} \vdash C}{\Gamma, \triangle A \vdash C}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A^{\div}} \qquad \frac{\Gamma, A \vdash C^{\div}}{\Gamma, A^{\div} \vdash C^{\div}}$$

especially if one says

$$\begin{array}{rcl} \text{True Props } A & ::= & \triangle D \mid \cdots \\ \text{Irrelevant Props } D & ::= & \star A \end{array}$$

and denotes the true judgment by superscript $\top$. Then the rules are

$$\frac{\Gamma \vdash D^{\div}}{\Gamma \vdash \triangle D^{\top}} \qquad \frac{\Gamma, D^{\div} \vdash A^{\top}}{\Gamma, \triangle D^{\top} \vdash A^{\top}}$$

$$\frac{\Gamma \vdash A^{\top}}{\Gamma \vdash \star A^{\div}} \qquad \frac{\Gamma, A^{\top} \vdash D^{\div}}{\Gamma, \star A^{\div} \vdash D^{\div}}$$

and are perfectly symmetric under interchanging $(\star, A, \div) \Leftrightarrow (\triangle, D, \top)$.

The cut-elimination proof for this system eluded me before, but now I think I have it. In heterogeneous cuts like $\Gamma \vdash A$ against $\Gamma, A \vdash C^{\div}$ and $\Gamma \vdash A^{\div}$ against $\Gamma, A^{\div} \vdash C$ analyze only the *second* derivation (the hypothetical one) for principal cuts could not even occur due to heterogeneity.

This *requires* (felicitously?) that left-rules for ordinary connectives require truth on the right. We could also define connectives native to the proof-irrelevant judgment, which would similarly require proof-irrelevance on the right.

2008.7.12

The translation also seems to work if I stick everything at weak and pepper with FU, or stick everything at strong and pepper with UF — not sure what happens to triangle, then. I thought I had some one-sided argument to permute some tof the left rules outside the bimonadic phase shift but I can't see how to apply it. Is the more left-rule-friendly system actually cut-eliminable? It's weird to see how the cut principle is 'different'

because of global properties of connectives in one system (namely that the judgments always match up) which reminds me of how pfenning-style proof irrelevance differed from awodey-bauer — which in turn sort of makes sense.

2008.7.13

Think about this: start with the Jul 2 mixed-strength translation. Reencode everything into the strong judgment by substituting, for example, $\uparrow(\downarrow A \vee \downarrow B)$ for $A \vee B$, and $(\downarrow A) \Rightarrow B$ for $A \Rightarrow B$, and $\uparrow\triangle\downarrow A$ for $\triangle A$. This should not affect provability.

So in the resulting translation, we get $U(FUFA \vee FUFB)$ for $A \vee B$, and $UFA \Rightarrow UFB$ for $A \Rightarrow B$, and $UFU'F'UFA$ for $\triangle A$. Observe that everything in this translation has the property that $UFX = X$. So we should really get away with $UF(A \vee B)$ and $A \Rightarrow B$ and $UFU'F'A$. So we're back at yes, really, monadizing all the positive things and bimonading the irrelevance modality should be faithful – well, additionally, I suppose I am applying F to every conclusion.

2008.7.14

Okay, there are only so many reasonable translations of a given connective. Under a translation where all weak things in the context are strengthened by U and all strong conclusions are weakened by F, the weak translation of $\overline{A} \Rightarrow \overline{B}$ as $F\overline{A} \multimap F\overline{B}$ is at least faithful. The variations of this weak translation with different inputs are all interdefinable:

$$F\overline{A} \multimap F\overline{B}$$

$$F\overline{A} \multimap FU\underline{B}$$

$$FU\underline{A} \multimap F\overline{B}$$

$$FU\underline{A} \multimap FU\underline{B}$$

If we hit these on the outside with U we get the four

$$\overline{A} \Rightarrow UF\overline{B}$$

$$\overline{A} \Rightarrow U\underline{B}$$

$$U\underline{A} \Rightarrow UF\overline{B}$$

$$U\underline{A} \Rightarrow U\underline{B}$$

If we hit these with F we get

$$F(\overline{A} \Rightarrow UF\overline{B})$$

$$F(\overline{A} \Rightarrow U\underline{B})$$

$$F(U\underline{A} \Rightarrow UF\overline{B})$$

$$F(U\underline{A} \Rightarrow U\underline{B})$$

And if we hit these with U we get back to the second row.

The picture with disjunction and irrelevance looks like

$$\begin{array}{lll} F\overline{A} \multimap F\overline{B} & U\underline{A} \vee U\underline{B} & \\ F\overline{A} \multimap FU\underline{B} & U\underline{A} \vee UF\overline{B} & U'F'U\underline{A} \\ FU\underline{A} \multimap F\overline{B} & UF\overline{A} \vee U\underline{B} & U'F'UF\overline{A} \\ FU\underline{A} \multimap FU\underline{B} & UF\overline{A} \vee UF\overline{A} & \end{array}$$

* * * * * *

$$\begin{array}{lll} \overline{A} \Rightarrow UF\overline{B} & FU\underline{A} \oplus FU\underline{B} & \\ \overline{A} \Rightarrow U\underline{B} & FU\underline{A} \oplus F\overline{B} & FU'F'U\underline{A} \\ U\underline{A} \Rightarrow UF\overline{B} & F\overline{A} \oplus FU\underline{B} & FU'F'UF\overline{A} \\ U\underline{A} \Rightarrow U\underline{B} & F\overline{A} \oplus F\overline{B} & \end{array}$$

* * * * * *

$$\begin{array}{lll} F(\overline{A} \Rightarrow UF\overline{B}) & U(FU\underline{A} \oplus FU\underline{B}) & \\ F(\overline{A} \Rightarrow U\underline{B}) & U(FU\underline{A} \oplus F\overline{B}) & UFU'F'U\underline{A} \\ F(U\underline{A} \Rightarrow UF\overline{B}) & U(F\overline{A} \oplus FU\underline{B}) & UFU'F'UF\overline{A} \\ F(U\underline{A} \Rightarrow U\underline{B}) & U(F\overline{A} \oplus F\overline{B}) & \end{array}$$

2008.7.15

Let $A^* = \bigcirc A^\times$. Define

$$\begin{array}{rcl} (A \star B)^\times & = & A^* \star B^* \\ (\triangle A)^\times & = & \bigcirc' A^* \\ p^\times & = & p \\ (A \star B)^\cdot & = & A^\cdot \star B^\cdot \\ (\triangle A)^\cdot & = & \bigcirc \bigcirc' A^\cdot \\ p^\cdot & = & p \end{array}$$

The theorem to prove (or refute!) is

Theorem 0.35 $\Gamma^\cdot \vdash A^\cdot$ *iff* $\Gamma^\times \vdash A^\times$

2008.7.16

Try the following generalization. We say $A \mapsto A^\cdot$ according to

$$\frac{A \mapsto A^\cdot}{\triangle A \mapsto \bigcirc \bigcirc' A^\cdot} \qquad \frac{A \mapsto A^\cdot}{\triangle A \mapsto^R \bigcirc' A^\cdot \,\mathrm{lax}}$$

$$\frac{A \mapsto A^{\cdot}}{\triangle A \mapsto \bigcirc' A^{\cdot}} \qquad \frac{A \mapsto A^{\cdot}}{\triangle A \mapsto^{R} A^{\cdot}\,\mathrm{lax}'}$$

$$\frac{A_1 \mapsto A_1^{\cdot} \qquad A_2 \mapsto A_2^{\cdot}}{A_1 \star A_2 \mapsto A_1^{\cdot} \star A_2^{\cdot}} \qquad \frac{}{p \mapsto p}$$

Theorem 0.36 *If* $\Gamma \vdash A \mapsto \Gamma^{\cdot} \vdash A^{\cdot}$ *and* $\Gamma^{\cdot} \vdash A^{\cdot}$, *then* $\Gamma^{\times} \vdash A^{\times}\ \mathrm{lax}$.

Proof By induction on the derivation.

Case:

$$\frac{\Gamma^{\cdot} \vdash \bigcirc' A^{\cdot}\,\mathrm{lax}}{\Gamma^{\cdot} \vdash \bigcirc \bigcirc' A^{\cdot}} \mapsto \frac{i.h.}{\Gamma^{\times} \vdash \bigcirc' A^{*}\,\mathrm{lax}}$$

Case:

$$\frac{\Gamma^{\cdot} \vdash \bigcirc' A^{\cdot}}{\Gamma^{\cdot} \vdash \bigcirc' A^{\cdot}\,\mathrm{lax}} \mapsto \frac{i.h.}{\Gamma^{\times} \vdash \bigcirc' A^{*}\,\mathrm{lax}}$$

Case:

$$\frac{\Gamma^{\cdot} \vdash A^{\cdot}\,\mathrm{lax}'}{\Gamma^{\cdot} \vdash \bigcirc' A^{\cdot}} \mapsto \frac{i.h.}{\Gamma^{\times} \vdash \bigcirc' A^{*}\,\mathrm{lax}}$$

Case:

$$\frac{\Gamma^{\cdot} \vdash A^{\cdot}}{\Gamma^{\cdot} \vdash A^{\cdot}\,\mathrm{lax}'} \mapsto \frac{\dfrac{\dfrac{\dfrac{\dfrac{i.h.}{\Gamma^{\times} \vdash A^{\times}}}{\Gamma^{\times} \vdash A^{\times}\,\mathrm{lax}}}{\Gamma^{\times} \vdash A^{*}}}{\Gamma^{\times} \vdash A^{*}\,\mathrm{lax}'}}{\Gamma^{\times} \vdash \bigcirc' A^{*}}}{\Gamma^{\times} \vdash \bigcirc' A^{*}\,\mathrm{lax}}$$

Case:

$$\frac{\Gamma^{\cdot}, \bigcirc' A^{\cdot} \vdash B^{\cdot}\,\mathrm{lax}}{\Gamma^{\cdot}, \bigcirc \bigcirc' A^{\cdot} \vdash B^{\cdot}\,\mathrm{lax}} \mapsto \frac{i.h.}{\Gamma^{\times}, \bigcirc' A^{*} \vdash \bigcirc' B^{*}\,\mathrm{lax}}$$

Case:

$$\frac{\Gamma^{\cdot}, A^{\cdot} \vdash B^{\cdot}\,\mathrm{lax}'}{\Gamma^{\cdot}, \bigcirc' A^{\cdot} \vdash B^{\cdot}\,\mathrm{lax}'} \mapsto \frac{\Gamma^{\times}, A^{\times} \vdash \bigcirc' B^{*}\,\mathrm{lax}\,\Gamma^{\times}, \bigcirc' A^{*} \vdash \bigcirc' B^{*}\,\mathrm{lax}}{}$$

This case seems to be broken.

Another tack.

Consider (i) the logic with a 'subordinate' modality like the monad (in contrast to the 'superordinate' comonad box) but where decomposing just that modality is disallowed to be decomposed on the left when the weak judgment is on the right, and (ii) the logic where *all* (truth-native) left rules are forbidden at the weak judgment.

I'm pretty sure I've convinced myself (*ii*) is analytic, and I suspect (*i*) is, too. They ought to correspond roughly to the $\times$ and $\cdot$ translations above.

Can I prove them equivalent directly by permuting rules outside of the weak judgment?

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Gamma, B \vdash C\,\mathrm{lax}}{\Gamma, B \vdash \triangle C}}{\Gamma, A \Rightarrow B \vdash \triangle C} \longleftrightarrow \dfrac{\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C\,\mathrm{lax}}{\Gamma, A \Rightarrow B \vdash C\,\mathrm{lax}}}{\Gamma, A \Rightarrow B \vdash \triangle C}$$

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Gamma, B \vdash C}{\Gamma, B \vdash C\,\mathrm{lax}}}{\Gamma, A \Rightarrow B \vdash C\,\mathrm{lax}} \longleftrightarrow \dfrac{\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \Rightarrow B \vdash C}}{\Gamma, A \Rightarrow B \vdash C\,\mathrm{lax}}$$

$$\dfrac{\Gamma, C^{\div} \vdash A \qquad \dfrac{\Gamma, B, C \vdash D\,\mathrm{lax}}{\Gamma, B, C^{\div} \vdash D\,\mathrm{lax}}}{\Gamma, A \Rightarrow B, C^{\div} \vdash D\,\mathrm{lax}} \longrightarrow \dfrac{\dfrac{\Gamma, C \vdash A \qquad \Gamma, B, C \vdash D\,\mathrm{lax}}{\Gamma, A \Rightarrow B, C \vdash D\,\mathrm{lax}}}{\Gamma, A \Rightarrow B, C^{\div} \vdash D\,\mathrm{lax}}$$

I only really want to apply the last two of these, and only in the left-to-right direction (which is the only way that works for the third one anyway) By assumption I have ripped out all the left rules from the derivation above, so my case analysis becomes effective.

For disjunction I can do

$$\dfrac{\dfrac{\Gamma, A \vdash C}{\Gamma, A \vdash C\,\mathrm{lax}} \quad \dfrac{\Gamma, B \vdash C}{\Gamma, B \vdash C\,\mathrm{lax}}}{\Gamma, A \vee B \vdash C\,\mathrm{lax}} \longrightarrow \dfrac{\dfrac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}}{\Gamma, A \vee B \vdash C\,\mathrm{lax}}$$

$$\dfrac{\Gamma, A, C^{\div} \vdash D\,\mathrm{lax} \quad \Gamma, B, C^{\div} \vdash D\,\mathrm{lax}}{\Gamma, A \vee B, C^{\div} \vdash D\,\mathrm{lax}} \rightarrow \dfrac{\dfrac{\Gamma, A, C \vdash D\,\mathrm{lax} \quad \Gamma, B, C \vdash D\,\mathrm{lax}}{\Gamma, A \vee B, C \vdash D\,\mathrm{lax}}}{\Gamma, A \vee B, C^{\div} \vdash D\,\mathrm{lax}}$$

The case analysis here is a little delicate — we have to grind away all the $\div L$ before both branches of the disjunction are surely right rules.

$\triangle$ *remains* left-prohibited above the lax judgment, so we need not transform it. I suppose I have to consider sequents with more promoted stuff to be smaller.

The bimonadic logic is funnily restricted in that in order to peel off the first layer of proof irrelevance on the left, we have to be at it on the right.

2008.7.17

It's interesting that Twelfing the monad is easy, but box is tough; because we have enough of a handle on the conclusion that erasing it is easy, but not the context.

Still, I think suddenly that I should be able to twelf the proof of the bimonadic/proof irrelevance thing. The progressive definition of proof irrelevance and its proof of cut elimination should be feasible in principle, and a good check that I thought about coverage correctly.

2008.7.18

The proof should work merely by commutation of ordinary rules *late* relative to the monadic phase, and commutation *early* of promotion from the regular judgment to monadic.

2008.7.19

Ok, so consider the system defined by

$$\frac{\Gamma \vdash A^{\div}}{\Gamma \vdash \triangle A} \qquad \frac{\Gamma, A^{\div} \vdash C}{\Gamma, \triangle A \vdash C}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A^{\div}} \qquad \frac{\Gamma, A \vdash C^{\div}}{\Gamma, A^{\div} \vdash C^{\div}}$$

and left rules for other connectives that require *truth* on the right.

This should satisfy the cut elimination principle

$$\text{If } \Gamma \vdash A^{*} \text{ and } \Gamma, A^{*} \vdash C^{\star}, \text{ then } \Gamma \vdash C^{\star}.$$

As noted on 2008.7.11.

This is the logic that is most transparently equivalent to stanard proof irrelevance, call it $\vdash_P$. Certainly if $\Gamma \vdash_P A$, then $\Gamma \vdash A$ — I just run through promotion step by step and get out. I can also show by an easy induction that

- if $\Gamma \vdash A$ then $\Gamma \vdash_P A$
- if $\Gamma \vdash A^{\div}$ then $\Gamma^{\oplus} \vdash_P A$

We can imagine a derivation system $\vdash_L$ (which is not obviosuly cut-free) where left rules for most connectives are allowed to fire no matter what the judgmnet on the right. Certainly this system has at least as many derivations as $\vdash$, but it also has no more. For we can permute left rules past promotion until they exit the irrelevant phase.

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Gamma, B \vdash C}{\Gamma, B \vdash C^{\div}}}{\Gamma, A \Rightarrow B \vdash C^{\div}} \quad \longrightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \Rightarrow B \vdash C}}{\Gamma, A \Rightarrow B \vdash C^{\div}}$$

$$\dfrac{\Gamma, C^{\div} \vdash A \qquad \dfrac{\Gamma, B, C \vdash D^{\div}}{\Gamma, B, C^{\div} \vdash D^{\div}}}{\Gamma, A \Rightarrow B, C^{\div} \vdash D^{\div}} \quad \longrightarrow \quad \dfrac{\dfrac{\Gamma, C \vdash A \qquad \Gamma, B, C \vdash D^{\div}}{\Gamma, A \Rightarrow B, C \vdash D^{\div}}}{\Gamma, A \Rightarrow B, C^{\div} \vdash D^{\div}}$$

Actually, wait; in the system $\vdash_L$ we still might as well promote all irrelevant hypotheses as soon as we can. And then we can prove a lemma

Lemma 0.37 *If* $\Gamma^{\oplus} \vdash_L A^{\div}$*, then* $\Gamma^{\oplus} \vdash_L A$

because no rule other than $\triangle L$ (which still requires truth on the right) introduces $\div$ on the left, which is the only thing that can take advantage of $\div$ on the right.

The other part is to see that $\vdash_L$ is equivalent to $\vdash_M$, in which by definition $\triangle A$ on the left is only allowed to be decomposed against $\triangle A$ on the right.

2008.7.20

To see that is relatively easy, for we do transformations that push $\triangle L$ later.

$$\dfrac{\dfrac{\Gamma, C^{\div} \vdash A \qquad \Gamma, B, C^{\div} \vdash D}{\Gamma, A \Rightarrow B, C^{\div} \vdash D}}{\Gamma, A \Rightarrow B, \triangle C \vdash D} \quad \longrightarrow \quad \dfrac{\dfrac{\Gamma, C^{\div} \vdash A}{\Gamma, \triangle C \vdash A} \quad \dfrac{\Gamma, B, C^{\div} \vdash D}{\Gamma, B, \triangle C \vdash D}}{\Gamma, A \Rightarrow B, \triangle C \vdash D}$$

$$\dfrac{\dfrac{\Gamma, C^{\div}, A \vdash B}{\Gamma, C^{\div} \vdash A \Rightarrow B}}{\Gamma, \triangle C \vdash A \Rightarrow B} \quad \longrightarrow \quad \dfrac{\dfrac{\Gamma, C^{\div}, A \vdash B}{\Gamma, \triangle C, A \vdash B}}{\Gamma, \triangle C \vdash A \Rightarrow B}$$

Now we want to show that $\triangle$ in this system is effectively $\bigcirc\bigcirc'$. Say $X^{\bigcirc}$ is defined by $(\triangle A)^{\bigcirc}$ with $\bigcirc \bigcirc' (A^{\bigcirc})$, and $(A^{\div})^{\bigcirc}$ with $\bigcirc'(A^{\bigcirc})$, and all else homomorphic.

Lemma 0.38

- *If* $\Gamma^{\bigcirc} \vdash A^{\bigcirc}$*, then* $\Gamma \vdash_M A$*.*
- *If* $\Gamma^{\bigcirc} \vdash \bigcirc'(A^{\bigcirc})\,\mathrm{lax}$*, then* $\Gamma \vdash_M \triangle A$*.*
- *If* $\Gamma^{\bigcirc} \vdash A^{\bigcirc}\,\mathrm{lax}'$*, then* $\Gamma \vdash_M A^{\div}$*.*

Proof By induction. ■

Lemma 0.39

- *If* $\Gamma \vdash_M A$*, then* $\Gamma^{\bigcirc} \vdash A^{\bigcirc}$*.*
- *If* $\Gamma \vdash_M \triangle A$*, then* $\Gamma^{\bigcirc} \vdash \bigcirc'(A^{\bigcirc})\,\mathrm{lax}$*.*
- *If* $\Gamma \vdash_M A^{\div}$*, then* $\Gamma^{\bigcirc} \vdash A^{\bigcirc}\,\mathrm{lax}'$*.*

Proof By induction, taking especial advantage of the right-invertibility of $\bigcirc$ and $\bigcirc'$. ■

2008.7.21

I find it nonetheless frustrating that I still don't know any way to phrase these rule-permutation arguments as cut derivations.

2008.7.22

Science is rejection of proof by authority at an institutional level; can this work at a personal level? How often should one distrust one's own memories and go out and experiment again?

2008.7.23

Reading some fascinating notes by Conway and collaborators about curvature. The nice thing about it is its description of several examples of things that wind up being essentially about the Euler characteristic of a polyhedron/facial graph.

Specifically: the angle deficit at a vertex is kind of like the curvature there. If we consider the dual polyhedron projected as a spherical polygon onto the 'celestial sphere' then the *area* of the face corresponding to each original vertex is the same as the angle deficit — so of course the total angle deficit is constant for topologically spherical graphs.

2008.7.24

Let a platonic solid be given, with n faces each of which is an m-gon, of which k join at each vertex.

It has Euler characteristic $2 = F - E + V = n - (nm/2) + (nm/k) = n - nm(1/2 - 1/k)$. Solving for n we get

$$n = \frac{2}{1 - m(1/2 - 1/k)} = \frac{4k}{2k - m(k-2)}$$

We want $k > 2$, $m > 2$, $n > 1$.

If $k = 3$, then we're looking for an m such that $6 - m$ divides 12. Thus m could be 3 (tetrahedron) or 4 (cube) or 5 (dodecahedron).

If $k = 4$, then we're looking for an m such that $8 - 2m$ divides 16. Thus m could be 3 (octahedron).

If $k = 5$, then we're looking for an m such that $10 - 3m$ divides 20. Thus m could be 3 (icosahedron).

Suppose $k \geq 6$ and that

$$\frac{4k}{2k - m(k-2)}$$

has an integer solution.

$$m = \frac{k}{k-2}\frac{2n-4}{n}$$

Now $k/(k-2)$ shrinks as k grows; so if $k \geq 6$, $k/(k-2) \leq 6/4$. Also $\frac{2n-4}{n}$ is strictly bounded above by 2. So if $k \geq 6$ then $m < 2(6/4) = 3$.

2008.7.25

The incidence algebra on a locally finite poset is the vector space of functions that assigns scalars to intervals. The algebra structure is a convolution-like multiplication

$$(f * g)(a, b) = \sum_{a \leq x \leq b} f(a, x)g(x, b)$$

Associativity comes from

$$\begin{aligned} & f * (g * h) \\ &= \sum_{a \leq x \leq b} f(a, x) \sum_{x \leq y \leq b} g(x, y)h(y, b) \\ &= \sum_{a \leq x \leq y \leq b} f(a, x)g(x, y)h(y, b) \\ &= \sum_{a \leq y \leq b} \left(\sum_{a \leq x \leq y} f(a, x)g(x, y) \right) h(y, b) \end{aligned}$$

$$= (f * g) * h$$

2008.7.26

Could substitute a much less flexible mechanism for monotonicity (say, construction out of $\multimap^n$ for $n \geq 1$) just for the thesis work, as long as it satisfies the appropriate property out at the end, and mention that a more sophisticated analysis might also be possible.

2008.7.27

Talked with gwillen a bit about socially-negotiated ontology stuff. I realize that the only remotely constructive ideas I've had about it resemble some old multiple-hash scheme I was told about back at Whizbang that I can't find on the internet for its brutal simplicity.

2008.7.28

It seems mostly like transitivity is a big issue in voting-modulo for equality; is it the only major one? Can I get any insight from 'local poset' topology? It also violates global transitivity while preserving it locally. The skeleton of a Moebius strip is kind of like a local equivalence relation that locally has two equivalence classes.

2008.7.29

A possible story for why $\triangle$ is a connective (or at least is so as much as $\bigcirc$, with some synchronous moves postponed as judgments) is that its rules consist only of 'free moves' on top of those licensed by focusing: not exactly asynchronous decomposition, but sequences that eventually result in at-least-as-strong sequents.

The example of such a move being dangerously darting into a strong judgment on the right or a weak judgment on the left, in order to immediately (and asynchronously) get back out to safety.

2008.7.30

Debugging Twelf unification. Questions:

- Where does pruning actually happen?
- At some point I will want to apply an inverse substitution and actually fail if it attempts underscores at any point. Is this incompatible with the usual system of delayed substitutions? It seems that I need to really check if that variable is used in the type of any other evar. How do I collect all evars if that's the case? Is this code already implemented? I seem to recall Frank said it was in a separate body of code that just did matching.

2008.7.31

Any collaborative system that conveniently allows the creation of taxonomies must as a special case allow it for a single person; and as an even more special case for a single person over a brief period of time.

These taxonomies have competing requirements of 'fuzziness' and 'crispness'. This seems reminiscent of a very Suberian competition between freedom and regulation. Also of competition in APIs and interfaces in programming — a more general, more flexible thing is not strictly more desirable.

2008.8.1

Platform games often have puzzles where there are a bunch of widgets that act in some periodic way, and you have to dash through them to avoid being hit by them. Is there any interest in a mechanism that does the opposite? Where you need to *hit* enough of them to succeed? There is some sense in which this question risks being meaningless; that the only thing that matters is that each element is 'in the right phase' and there's no meaningful extrinsic notion of what counts as a 'hit' or 'miss'.

2008.8.2

I want to account for what I believe.

Setting it down as text and examining it seems like a fruitful thing to do.

Why is this so specifically in this case? It is because it is generally.

It seems that mere *beliefs* are somehow promoted to *knowledge* by virtue of their *cumulativity*.

This is why scientific thinking is fruitful.

What is meant by a belief anyhow? Some kind of abstraction of the internal state of an agent living in the world. A piece of data of some kind.

It is *correct* inasmuch as it is a useful piece of data in the model, one that produces good results. We can pull back any normative notions we have about the world back to normative notions about beliefs.

An agent appears to be learning (in a good sense) if exposure to more data about the world leads to better beliefs. Amazing how much normative thinking is coded here!

2008.8.3

There is such a thing as 'knowing english', at least to the extent that if a hungry person is dropped in the middle of an american city with a pile of money, one such will obtain food faster than others.

There is such a thing as 'knowing how to build an airplane' (knowledge that is spread across a group of people in a society) that can be distinguished

by whether airplanes actually get built or not, presuming a desire for such.

These are pretty much indisputable: knowledge, science are cumulative at least to this extent. You can't simply be told to shift sideways into a belief-state of knowing english, or being able to build airplanes. These beliefs, these bits of data, are *generally*, as an *approximation*, additive with other pieces of knowledge, but this may break down. *Knowing* english may (or may not) interfere with other languages.

2008.8.4

A finite sequence $(v_i)_{i\in I}$ is a *transaction* over I if $\sum_{i\in I} v_i = 0$. A transaction is of cardinality n if $|I| = n$. It is a *primitive* transaction if it is zero everywhere except for two coordinates i and j such that $v_i = -v_j$. The notation $[x]^i_j$ means the primitive transaction v where $v_i = -x$ and $v_j = x$.

Question For a given transaction of size n, what is the smallest number of primitive transactions that sum to it?

A simple upper bound is $n-1$. Suppose wlog that $I = \{1,\dots,n\}$. Then take the sequence of primitive transactions $w_k = \left[\sum_{i=1}^{k} v_i\right]_k^{k+1}$ for $k \in \{1,\dots,n-1\}$.

If a transaction v over I is *simple* if it has no nontrivial subtransactions in the evident sense — if it lacks any subset $J \subset I$ such that $\sum_{j\in J} v_j = 0$. For simple transactions the upper bound mentioned above is tight. If the graph induced by the primitive transactions used to recover v is not connected, then it has connected components, each of which is a subtransaction. So it must be connected — but we have to use at least $n-1$ edges to connect n vertices.

We can see therefore that any putative minimum can at least be reduced to a collection of minimum solutions for connected components of the primitive transaction graph — any cycles can be easily broken.

The question reduces to

Question For a given transaction of size n, what is the maximum subtransaction cover?

2008.8.5

The above is easily seen to be NP-hard. Reduce from the knapsack problem: if we have weights w_i that sum to $2m$, let there be $|I|$ people that owe w_i and 2 people who are each owed m. If there is a subset of the w_i that add to exactly m, we can do it in $|I|$ transactions, otherwise it takes $|I|+1$.

2008.8.6

The thing that perplexes me with typographers' way of talking about their obsession with whitespace and artists' way of talking about volumes and outlines and forms and so on is that it overdetermines things.

The set of pixels that are black and the set that are white are both determined by the underlying function $R^2 \to bool$ — you can't choose one without choosing (or in other cases, at least influencing) the other.

I suppose the reality they're talking about (or could be talking about) is human error functions: if I *focus on* the whitespace, whatever that means, then I might make a different result.

2008.8.7

In particular, to draw a walking figure, I must draw the whole figure. But a different figure results — empirically — if I start by placing the head where I think it should go, or if I start by placing the feet where I think they should go. This being because where I think other parts should go afterwards is influenced by earlier decisions.

2008.8.8

If I were perfectly accurate in all these estimates, order wouldn't matter — can this be taken as a *definition* of fidelity/accuracy when there is no objective exemplar of correctness to compare to?

2008.8.9

This reveals a hypocrisy in my thinking in the past. I've tried to argue to myself that the only long-term solution to spam is whitelisting.

That is, I'm claiming that we will fail as long as we basically accept every message but reject some, and that we might succeed by basically rejecting every message but accept some.

However, in both cases, we have a function which selects which messages are accepted. To be saying something interesting and nontrivial I must constrain what this function looks like.

Likewise with accepting and rejecting programs with type systems. We *want* to say that we should essentially start by rejecting all programs and then accept some, but what *should* we say if we want to be clear?

Likewise naïve set theory, although here we seem to have a sharper problem. For we can't consistently achieve comprehension of the universe in order to make the argument that it's just *some* and just the same (extensional) subset of the universe no matter how it's (intensionally) conceived. *Constructing* sets from axioms seems to be the only method that works.

Does the 'adjoint' perspective on modalities help any with reasoning

about classical modal logic's equivalence to intuitionistic logic? With understanding why it's difficult to come up with a labelled sequent calculus for Pfenning-Davies box?

How could something like Twelf be more 'self-eating'?

Turns out the last step of the reduction of the little transaction toy problem above is totally standard; subset sum, even when the target sum is zero, is classically NP-complete. Demanding a strict subset only causes a linear blow-up.

Added expandSub to unify.fun, and got rid of a case where Undef appearing in invertSub causes a NotInvertible exception. Less sure the latter is correct. Also feeling sketchy that there seems to be another expand-as-necessary feature in invertSub in addition to expandSub, which operates on a different context size.

It does look like I can 'explain' a labelled sequent calculus for Pfenning-Davies box.

There are two translations, A_p^+ and A_p^-. Both take as input propositions over the modal propositional langage. They output propositions in a language with a truth judgment and an index set of lax judgments all beneath it. A_p^+ yields a proposition suitable for the lax_p judgment, and A_p^- yields an ordinary proposition. We write truth connectives as $\Rightarrow, \vee, \wedge, etc$ and lax_p connectives as $\multimap_p, \oplus_p, \&_p, etc$. The adjoints U_p and F_p mediate between them.

We can also include an @ connective.

The translations are

X	X_p^-	X_p^+
a	$U_p a$	a
$A \Rightarrow B$	$U_p A_p^+ \Rightarrow U_p F_p B_p^-$	$F_p A_p^- \multimap B_p^+$
$A \vee B$	$U_p F_p (A_p^- \vee B_p^-)$	$F_p U_p A_p^+ \oplus F_p U_p B_p^+$
$A \wedge B$	$U_p F_p (A_p^- \wedge B_p^-)$	$F_p U_p A_p^+ \otimes F_p U_p B_p^+$
$\Box A$	$U_p F_p \forall \alpha . A_\alpha^-$	$F_p \forall \alpha . U_\alpha A_\alpha^+$
$A@q$	$U_p F_p A_q^-$	$F_p U_q A_q^+$

We can prove that the two translations are essentially similar up to provability

$$U_p A_p^+ \dashv\vdash A_p^-$$

but although the negative translation can up to provability be defined in terms of the positive, it is convenient in the form it's in for connecting it to the labelled Pfenning-Davies system.

A sequent $A_1[p_1], \ldots, A_n[p_n] \vdash C[q]$ gets translated to

$$(A_1)^-_{p_1}\mathtt{t}, \ldots, (A_n)^-_{p_n}\mathtt{t} \ \vdash C^+_q \, \mathrm{lax}_q$$

In reasoning about the translation we must sequence the implication with the second U_p in $U_pA^+_p \Rightarrow U_pF_pB^-_p$, and the F_ps with $\oplus$ in $F_pU_pA^+_p \oplus F_pU_pB^+_p$ and similarly with $\otimes$.

* * * * * *

In the negative translation of $\Box$, we could also do

$$U_pF_p\forall\alpha.U_\alpha F_\alpha A^-_\alpha$$

which changes the interpretation of $A[*]$ to mean that we can instantiate $A[*]$ at $A[q]$ only if q is already on the right. The reason why it doesn't matter whether this restriction is in or not is the fact that $U_\alpha F_\alpha A^-_\alpha$ and A^-_α are equivalent.

* * * * * *

Diamond also works:

X	X^-_p	X^+_p
$\Box A$	$U_pF_p\forall\alpha.U_\alpha F_\alpha A^-_\alpha$	$F_p\forall\alpha.U_\alpha A^+_\alpha$
$\Diamond A$	$(\forall\alpha.(A^-_\alpha \Rightarrow U_\alpha b_\alpha)) \Rightarrow U_p b_p$	$F_p(\forall\alpha.(U_\alpha A^+_\alpha \Rightarrow U_\alpha b_\alpha)) \multimap b_p$

where b is an indexed linear negative atom.

2008.8.10

I think cut elimination can be proved directly for the above system if one takes care to think of left-principal derivations separately.

Something like $\Gamma; A \vdash C[p]$ meaning essentially $\Gamma, A[p] \vdash C[p]$ and that A was most recently decomposed. Have a rule

$$\frac{\Gamma; A \vdash C[p]}{\Gamma, A[p] \vdash C[p]}$$

and put left rules at it, e.g.

$$\frac{\Gamma, A[p], B[p] \vdash C[p]}{\Gamma; A \wedge B \vdash C[p]}$$

The cut principles are then something like

$$\frac{\Gamma \vdash A[p] \qquad \Gamma, A[p] \vdash C[r]}{\Gamma \vdash C[r]}$$

$$\frac{\Gamma \vdash A[p] \qquad \Gamma; A \vdash C[p]}{\Gamma \vdash C[p]}$$

2008.8.11

Frank seemed to think that Deepak's system already has left-rule judgment constraints?

Goals for this coming week: reread carsten's thesis again. Work on implementation. Ideally have unification working, type reconstruction kind of working.

2008.8.12

I think I can curry away additives — this is a surprising fact that depends on having HLF around.

Probably generalizing monotonicity to that and n-ary arrow would actually be fairly flexible.

I can imagine a sci-fi story that operates on the premise that brain-states can be observed and interpreted fairly accurately, and the discovery is made that dreams are actually less narrative than we think, that they are only random jumblings of thoughts and symbols, and any narrative we "remember" in the morning is merely that, a epiphenomenal memory-experience that does not terribly accurately represent even the internal history of our own brains during the course of the night.

A new sport for the OCD set: tile-walking. Essentially a form of dance, participants are judged on the aesthetics of how their walk appears relative to a square grid.

Actually this suggests a more 'spatial' DDR where step directions are integrated, possibly with momentum.

MLML?

So Deepak tells me that his 'K says A' is basically $\Box_K(K \Rightarrow A)$. This makes a lot of sense. It's remarkable how close my attempt at representing the K in $\Gamma \xrightarrow{K} A$ as a linear token came without being correct. Deepak

(or more to the point, the correct approach) is instead using *modal* erasure to enforce the uniqueness of the K rather than linearity, which leads to a subtle difference in how the operation behaves on the left and right. This explains why when using linearity I seemed caught between kind of wanting $K \multimap (K \otimes A)$ on one side and $K \otimes (K \multimap A)$ on the other.

2008.8.13

Ran tests on directory cut-elim. Fixed a few missing cases for underscore.

2008.8.14

Other tests:

Test	**New Unif**	**Old Unif**
`alloc-sem`	Freezing violation	〃
`arith`	Ok	〃
`ccc`	Ok	〃
`church-rosser`	Ok	〃
`compile/cls`	Ok	〃
`compile/cpm`	Ok	〃
`compile/cps`	Ok	〃
`compile/cxm`	Ok	〃
`compile/debruijn`	Ok	〃
`compile/debruijn1`	Ok	〃
`cpsocc`	Ok	〃
`cut-elim`	Already Ok	〃
`fj`	Ok	〃
`fol`	Ok	〃
`guide`	Ok	〃
`handbook`	Ok	〃
`incll`	Ok	〃
`js4`	Ok	〃
`kolm`	Double-checking failed?	Ok
`lp-horn`	Ok	〃
`lp`	Freezing error	〃
`mini-ml`	Ok	〃
`modal`	Missing sources.cfg	〃
`polylam`	Ok	〃
`prop-calc`	Ok	〃
`small-step`	Missing sources.cfg	〃
`tabled`	Didn't do	〃
`tapl-ch13`	Ok	〃

2008.8.15

So the `kolm` bug is real. It is in the midst of some code that actually runs the theorem *prover*. I wrote some code to copy the ref cells in a unificaion problem and ran old unification and new in parallel to test for discrepancies.

2008.8.16

Testing for discrepancies failed very mysteriously. Trying to make a minimal counterexample instead:

```
i : type.
o : type.
forall : (i -> o) -> o.
kolm : o -> type.
kolm_forall : kolm (forall A)
    <- ({a:i} kolm (A a)).
%theorem exkolm :  forallG (pi {a:i})
   forall {A:o} exists {K:kolm A} true.
%prove 3 A (exkolm A K).
```

2008.8.17

So the proof that proof irrelevance is equivalent to two monads is relatively easy after all.

Let $—^*$ be the transformation that rewrites $\triangle$ to $\bigcirc\bigcirc'$.

Lemma 0.40

- *If* $\Gamma^*, \bigcirc'\Gamma' \vdash A^*$, *then* $\Gamma, (\Gamma')^\div \vdash A$.
- *If* $\Gamma^*, \bigcirc'\Gamma' \vdash \bigcirc' A\ J$, *then* $\Gamma, (\Gamma')^\div \vdash \triangle A$ *(for* $J \in \{\mathrm{lax}, \mathrm{t}\}$*)*
- *If* $\Gamma^*, \bigcirc'\Gamma' \vdash A^*\ \mathrm{lax}'$, *then* $\Gamma, (\Gamma')^\div \vdash A^\div$.

Proof Analyze the derivation in the bimonadic logic. Right rules are generally easy, and proceed down the list of judgments when they actually involve the monads. Nonmonadic left rules are easy except in the last case, where we must prove lemmas like

Lemma 0.41 *If* $\Gamma, A \vdash C^\div$ *and* $\Gamma, B \vdash C^\div$, *then* $\Gamma, A \vee B \vdash C^\div$.

which are nonetheless rather easy. The monadic left rules only come up in two places, and are translated to triangle and irrelevance left rules. ■

2008.8.18

To do a labelled presentation of a modal system with $x < y < z$, I need a 'lexicographically ordered' set of labels, $x_{ij} < y_i < z$. The stronger box conjures up new i and j, whereas the weaker box just comes up with new i.

Ordinarily every connective dips down from the top all the way to the bottom; so ordinary truth requires both labels to match up on the right.

Possibly something like the following works:

X	X^-_{pq}	X^+_{pq}
$\Box_1 A$	$U_{pq} F^{pq} \forall \alpha\beta . U_{\alpha\beta} F^{\alpha\beta} A^-_{\alpha\beta}$	$F^{pq} \forall \alpha\beta . U_{\alpha\beta} A^+_{\alpha\beta}$
$\Box A$	$U_{pq} F^{pq} U_p F^p \forall \alpha . U_{p\alpha} F^{p\alpha} A^-_{p\alpha}$	$F^{pq}_p \forall \alpha . U^p_{p\alpha} A^+_{p\alpha}$

2008.8.19

No, maybe the following:

X	X^-_{pq}	X^+_{pq}
$\Box_1 A$	$U_{pq} F^{pq} \forall \alpha\beta . U_{\alpha\beta} F^{\alpha\beta} A^-_{\alpha\beta}$	$F^{pq} \forall \alpha\beta . U_{\alpha\beta} A^+_{\alpha\beta}$
$\Box A$	$U_{pq} F^{pq} \forall \alpha . U_{p\alpha} F^{p\alpha} A^-_{p\alpha}$	$F^{pq} \forall \alpha . U_{p\alpha} A^+_{p\alpha}$

Synchronous sequencing the $\forall \alpha$ and the $U_{p\alpha}$ makes the weaker validity actually weaker — it only works if the world begins with p.

The point is that $\Box$ scrambles all labels below some point, or perhaps more to the point, those 'not above' — I probably can still think of the multi-labels as tuples rather than paths. The poset is embedded into the subset order on the powerset of the original graph.

I think if I'm thinking about the round-trip inherent in box, the preorder structure collapses and all I can really tell apart is a poset.

For $S \subseteq X$ I get something like

$$(\Box_S A)^-_{x:X} = U_x F^x \forall s{:}S . U_{x+s} F^{x+s} A^-_{x+s}$$

$$(\Box_S A)^+_{x:X} = F^x \forall s{:}S . U_{x+s} A^+_{x+s}$$

This seems to only require one flat domain of lax truth judgments below regular truth! How strange.

2008.8.19

I should maybe try to trace every essentially stateful thing inside of unification — put printfs at the evar instantiations and constraint creations, for instance.

2008.8.20

In sml, I always forget the 'type' in 'where type' and consequently get weird error messages. Another thing to remember is that earlier functor arguments are indeed in scope.

2008.8.21

Laura Marsh asked me a question about work shift scheduling that is (almost certainly) a lovely example of classic NP-complete problems showing up in real life.

2008.8.22

Have been going through and uncurrying the additives by hand. The additive conjunction left rule comes out as

```
andL : (hyp (A and B) -o conc C)
        o- ({a : w}{b : here}
            hyp A @ a
         -> hyp B @ a
         -> conc C @ a * b).
```

which funnily enough has a zero-ary analogue

```
topL : (hyp top -o conc C)
        o- ({a : w}{b : here} conc C @ a * b).
```

which is *definable* as

```
topL : (hyp top -o conc C)
        o- ({a : w}{b : here} conc C @ a * b)
 = [x] [y] x.
```

Oh, wait, yikes, that's not the andL I meant! The standard one would be

```
andL1 : (hyp (A and B) -o conc C)
        o- (hyp A -o conc C).
andL2 : (hyp (A and B) -o conc C)
        o- (hyp B -o conc C).
```

I wonder if the former also 'works', though, and if so, in what sense?

2008.8.23

I can't work out how I was inferring the existence of world-underscores before. The capital lambdas make sense, but not the omitted applications.

2008.8.24

Trying an alternate tack, reifying them instead — it seems to motivate doing unification with underscores, since then I can solve

$$u[] = v[x.x]$$

by projecting out *both* arguments of v.

2008.8.25

The more pertinent non-pattern problem is

$$u[] = v[x.k]$$

because I'm using an arbitrary constant to represent world application.

This I don't really know how to deal with. Surprisingly, changing the definition of pattern to allow specifically ^ doesn't seem to work. Maybe fiddling with subordination would?

2008.8.26

Some unification transformations for ACU.

$$\begin{gathered} v[D] = \bar{\alpha} * \bar{u}[\bar{D}] \\ \mapsto v \leftarrow \bar{\alpha} * \bar{u}'[\bar{D}\big|_D] \\ u_i \leftarrow u_i'[D_i\big|_D] \end{gathered}$$

(if $\forall j.\alpha_j \in D$)

$$\alpha * \bar{\beta} * \bar{u}[\bar{D}] = \alpha * \bar{\gamma} * \bar{v}[\bar{D}] \mapsto \bar{\beta} * \bar{u}[\bar{D}] = \bar{\gamma} * \bar{v}[\bar{D}]$$

$$\begin{gathered} \alpha * \bar{\beta} * \bar{u}[\bar{D}] = w[D] * \bar{\gamma} * \bar{v}[\bar{D}] \\ \mapsto \bar{\beta} * \bar{u}[\bar{D}] = w'[D] * \bar{\gamma} * \bar{v}[\bar{D}] \\ w[D] \leftarrow \alpha * w'[D] \end{gathered}$$

(if $\alpha \notin \bar{\gamma}, v[\bar{D}]$ and $\alpha \in D$)

2008.8.27

Actually, if all I care about is MGU solvability, why don't I just solve constraints over $\mathbb{Q}$ instead of $\mathbb{N}$?

Subordination doesn't work to get rid of spurious world constraints. I really think something more along the lines of modifying the notion of patterns is right.

2008.8.28

Frank suggested that $u[v[\alpha]] = \alpha \mapsto u \leftarrow 1 \wedge v \leftarrow 1$ should be generalized. I tend to like

$$\bar{\gamma} * u[p] = \alpha * \bar{\beta} \mapsto u \leftarrow 1 \wedge \bar{\gamma} * p = \alpha * \bar{\beta}$$

if $\alpha \notin \bar{\beta}, \bar{\gamma}$.

2008.8.29

Consider the possibility that 'truth' is defined by the structure of a given proposition or predicate, not any overriding notion of truth — that there might be nothing significant in common between "this is red" and "I like this", only that the predicates of redness and preference are satisfied.

2008.8.30

Suprisingly and upsettingly, reconstruction of the principal cut case for $\multimap$ seems to return a constraint of the form $a * b = c * d$.

2008.8.31

Oops! That constraint came from having written the case incorrectly and naming some variables differently that ought to have been the same.

The correct transform is not

$$u[v[n]] = n \mapsto u \leftarrow 1$$

but rather

$$u[v[n]] = n \mapsto u \leftarrow 1 * w[]$$

because the latter preserves open instantiations.

2008.9.1

Consider what to do with

$$u_1[\sigma_1] \cdots u_m[\sigma_M] = n * X$$

when n only occurs in, say, σ_j among the σ and does not occur in X.

Take specifically

$$u[1] * v[] = 1 * h[]$$

I'm tempted to transform $u \leftarrow 1$, but that's not correct for the same reason as the previous problem. I really want to make up a new evar j and say that $u \leftarrow 1 * j[]$.

2008.9.2

Here's my current smallest counterexample to `lincut.elf` working:

```
%hlf.

o : type. %name o A.
and    : o -> o -> o.  %infix right 11 and.
conc : o -> type.
hyp : o -> type.
andl1 : (hyp (A and B) -o conc C)
        o- (hyp A -o conc C).

ca : {a : w} {b : w} {A : o}
      conc A @ a
      -> (hyp A -o conc C) @ b
      -> conc C @ a * b
      -> type.

car_andl1:  {A:o} {B1:o} {A1:o} {D:conc A}
            {E1': hyp B1 -o conc A1} {A2:o}
            {H:hyp (B1 and A2)}
            {E1: hyp A -o hyp B1 -o conc A1}
    ca ^ ^ A D
            ([h:^(hyp A)] andl1 ^ (E1 ^ h) ^ H)
            (andl1 ^ E1' ^ H).
```

The constraints generated by `car_andl1` are easily seen to be unsolvable, so either there is a problem with the twelf code as written, or with my constraint generation code.

2008.9.3

This reduces to

```
%hlf.
o : type.
n : (o -o o) -o o.
p :  {b : w} (o -o o) @ b -> type.
c: {E: o -o o -o o} p ^ ([h:^ o] n ^ ([x :^ o] E ^ h ^ x)).
```

2008.9.4

Ah, the cause of the bug is the same old issue I always forget about when doing dependent types, that I must shift the types of looked-up variables when pulling them out for synthesis.

2008.9.5

There were some other bugs with zero-ary lollipop being too eager and other miscellaneous debruijn problems.

2008.9.6

Got most bugs fixed now, I think, and coded up the disequality of lambda terms.

2008.9.7

This example does fail during abstraction:

```
o : type.
c : ((o -o o) -o o) -> type.
k : c A.
```

Apparently there are some contextual variables left over. Too bad! I lack a clear picture still of what the example looks like when run having turned debugging off. Should get around to that soon.

2008.9.8

SVG cursors are kind of messed up in Chrome, especially hotpsots. Safari does them correctly apparently, but slows down more significantly if it has to draw a lot.

I probably have a minor bug when naming worlds.

The following works:

```
%hlf.

tm : type.
name : type.

lam : (name -0 tm) -> tm.
var : name -0 tm.
app : tm -> tm -> tm.

#' : name -0 name -0 type.
# = [a] [b] #' ^ a ^ b.
%infix none 5 #.

subst : tm -> name -0 tm -> tm -> type.
subst/app : {a:w}
     subst E ^ N (app M1 M2) (app M1' M2') @ a
     <- subst E ^ N M1 M1' @ a
     <- subst E ^ N M2 M2' @ a.

subst/lam : subst E ^ N (lam M) (lam M')
```

```
    o- ({n :^ name} subst E ^ N (M ^ n) (M' ^ n)).

subst/var/this : subst E ^ N (var ^ N) E.
subst/var/that : subst E ^ N (var ^ N') (var ^ N')
  <- N # N'.
```

but in `subst/app` for instance the implicit world gets named E.

I wonder what would happen in the pi-calculus, when one wants to use linearity for two different things? Might be okay still.

2008.9.9

Interesting pattern when trying to do type inference on an unη-expanded variable checked against $(\cdots(o \multimap o)\cdots \multimap o)$ at various orders.

The code to generate the unification problem and type for 4th order for example is

```
o : type.
c : (((o -o o) -o o) -o o) -> type.
d : c A.
```

and

```
o : type.
c : (((o -o o) -o o) -o o) -> type.
d : {(((o -o o) -o o) -o o)}  c A.
```

The results up to 7th order are:

```
2nd
===
((all.(1 -> 1)) -> e)
((all.(1 -> 1)) -> e)

3rd
===
((all.((all.(B[1.2] -> 2 * B[1.2])) -> 1)) -> e)
((all.((all.(1 -> 1 * 2)) -> 1)) -> e)
Solution:
B <- 1

4th
===
((all.((all.((all.(1 -> 1 * H[2.3])) -> 2 * H[1.2])) -> 1)) -> e)
((all.((all.((all.(1 -> 1 * 2)) -> 1 * 2)) -> 1)) -> e)
Solution:
H <- 1

5th
===
((all.((all.((all.((all.(B[1.2.3.4] -> C[1.2.3.4])) -> 1 * J[2.3])) -> 2 * J[1.2] * K[1.2])) -> 1)) -> e)
((all.((all.((all.((all.(1 -> 1 * 2)) -> 1 * 2)) -> 1 * 2)) -> 1)) -> e)
Solution:
J <- 1
K <- e
B <- 1
C <- 1 * 2

6th
```

```
===
((all.((all.((all.((all.((all.(1 -> 1 * M[2.3.4.5])) -> D[1.2.3.4])) -> 1 * L[2.3])) -> 2 * L[1.2] * O[1.2])) -> 1)) -> e)
((all.((all.((all.((all.((all.(1 -> 1 * 2)) -> 1 * 2)) -> 1 * 2)) -> 1 * 2)) -> 1)) -> e)
Solution:
M <- 1
D <- 1 * 2
L <- 1
O <- e

7th
===
((all.((all.((all.((all.((all.((all.(B[1.2.3.4.5.6] -> C[1.2.3.4.5.6])) -> 1 * O[2.3.4.5])) -> E[1.2.3.4])) -> 1 *
   N[2.3])) -> 2 * N[1.2] * P[1.2])) -> 1)) -> e)
((all.((all.((all.((all.((all.((all.(1 -> 1 * 2)) -> 1 * 2)) -> 1 * 2)) -> 1 * 2))  -> 1 * 2)) -> 1)) -> e)
Solution:
B <- 1
C <- 1 * 2
O <- 1
E <- 1 * 2
N <- 1
P <- e
```

The solutions are computed by hand.

New bug?

```
o : type.
c : ({a:w}{b:w} o @ a -> o @ b) -> type.
d : {a1:w}{a2:w} {X : {a3:w}{a4:w} o @ a1 -> o @ a2} c X.
```

Whoops! No, I had just turned off the debugging code that reported failure. I should deal with that better. Also the following (correctly) fails, contrary to discussion in the entry from 2007.9.21 above:

```
o : type.

c : ({a:w}{b:w} o @ a -> o @ a -> o @ b)
    -> ({a:w}{b:w} o @ a -> o @ b -> o @ b)
    -> type.

d : c X X.
```

A building that is explicitly linear, in which one must walk a space-filling curve. Which may have the occasional room. Which gets narrower as you approach the exit. The choiceless species of labyrinth. Each participant is given a permanent marker, red, green, or blue. Every surface of the space is white, invites marking — except that it is prepared with graffiti, hostile to, or agressively supportive of, “the reds”, “the greens”, “the blues”.

And the ceiling gets lower as you go on, too, and incidental objects in the room become smaller.

2008.9.10

I almost saw last night what Noam was getting at when he was trying to explain his funny context business — that some theorems might get shorter if you had the ability to see variable occurence in more than one way, that

$$\frac{\Gamma' = \Gamma(A) \Longrightarrow \Gamma' \vdash B}{\Gamma \vdash A \to B}$$

could be an *admissible* rule (and the ordinary rule admissible wrt it) for which, say, weakening might be easier to show without having to actually induct any farther.

Everything in the type inference results from yesterday makes sense until one gets to 5th order. For 3rd order, I see that X must have type

$\forall\alpha.(\forall\beta.u[\alpha\beta] \to \alpha * u[\alpha\beta]) \to \alpha$

but see that this is equivalent to

$\forall\alpha.(\forall\beta.\beta \to \alpha * \beta) \to \alpha$

because the β, being quantified at a negative position, can be instantiated so freely that it doesn't matter that there is an unknown $u : \mathsf{w} \to \mathsf{w} \to \mathsf{w}$ there.

I would like to be able to write a large program with many features that are effectively independently controlled by #ifdefs, but be able to tell nonetheless some static properties of the program (at a minimum syntactic correctness) under any of the 2^n possible 'configurations', and especially to derive source-language versions of the program without those features.

I think of this as especially useful when facing a big complicated program that could be simpler and better understood if only many useful features were removed; I don't want to ultimately use the simple version, but I'd like to see what it is and then compare it to the more complicated version.

In all likelihood looking at the development version history of a project would be a good approximation, but you would lose out on bugfixes and significant reorganiztion by looking only at early versions.

Fontforge compiles under cygwin with

```
./configure --without-python --disable-pyextension
            --without-freetype-src
```

2008.9.11

Read some of a Dexter Kozen paper on stochastic processes and coinduction. Didn't really grasp it.

2008.9.12

Went to Susmit's thesis defense. Very strange; he seemed to evade questions when it wasn't necessary.

2008.9.13

I think gaming is the right medium for expressing ideas about, e.g., societal engineering — the message is 'Here! Here is this dispassionate, detached simulation. Here are my assumptions (question them if you want) and here are the things you can modify and and here is the result.'

2008.9.14

The thing we expect to do with patterns for pair types actually falls out somewhat naturally from the pattern restriction on functions after uncurrying them; the notion disjoint projections of the same variable amounts to disjoint sets of variables after partial projections are η-expanded.

2008.9.15

A useful way to find out what resources a flash file is getting programmatically from some directory (even if decompiling fails) is to just point it to one's own server and look at the access logs.

'ChucK' is a decent, if not spectacular, imperative music programming environment.

2008.9.16

Is it really all that useful to have disequality for store names? It might make certain predicates easier to define, like the 'pattern-ness' of the store as a whole. For that I wouldn't even need sharp as a separate relation, just linearity itself. That would be rather nice to define, in fact.

I feel about mathematicians who say 'but we must have AC! we need it for the following important and useful results' much the same way as I do about religious people who say 'but we must have faith in God! for otherwise how would our souls enter Heaven?'

2008.9.17

Aha! At least I figured out why I was getting double-checking without enabling it.

Index: inference.fun

```
===================================================================
--- inference.fun (revision 1314)
+++ inference.fun (working copy)
@@ -154,7 +154,7 @@
    val ((Gnew, Bnew), sc) = expand' ((G, B), (G, B), 0)
    val _ = if (!Global.doubleCheck) then TypeCheck.typeCheckCtx (Gnew) else ()
    val ((G', B'), w') = sc ((Gnew, Bnew), I.id)
-  val _ = TypeCheck.typeCheckCtx G'
+   val _ = if (!Global.doubleCheck) then TypeCheck.typeCheckCtx G' else ()

    val S' = S.State (n, (G', B'), (IH, OH), d, S.orderSub (O, w'),
     map (fn (i, F') => (i, F.forSub (F', w'))) H, F.forSub (F, w'))
```

2008.9.18

Unfortunately I still cannot find the point of divergence of my code from the original. Perhaps it is due to the eta-expansion of substitutions after all? I have not checked whether the explicit eta-expansion is at fault — it may be happening elsewhere 'implicitly' in lazy expansion of certain case analyses.

It doesn't seem to be due to exceptional returns — that was tom's suggestion, and I was really hoping it was the reason.

2008.9.19

Least-squares fit to required gradient amounts to equality in Laplace space! Awfully cute, that.

2008.9.20

To get properly centered slides I need to run `dvips` like this:

```
dvips  -T 11in,8.5in slides
```

WMM talks:

Formalizing an Extensional Semantics for Units of Measure **Andrew Kennedy**

Invariance under scaling the important issue.

Kind of like representation independence in polymorphism and free theorems.

Use logical relations similarly to prove results.

$$g : \forall u.num\ u \to num\ (u^2)$$
$$\forall x.g(kx) = k^2(gx)$$

$$bar : \forall \alpha.\alpha \times \alpha$$
$$\forall x.bar(f(x)) = \langle f, f \rangle (bar(x))$$

I feel slightly confused that 'nice ex post facto results' are conflated with the definition of 'not going wrong'.

They said you can't do square root $u^2 \to u$ without a square root primitive — but certainly you can write newton's method $u^2 \to u \to u$ given an initial guess since $g := (t + g^2)/2g$ is still well-united. Is there any way of thinking about how this algorithm is invariant under scaling, up to approximation?

Is there anything to say about Tarantola's ideas about invariance up to choice of, e.g., m vs. m^{-1}?

Proving correctness of a dynamic atomicity analysis in Coq

Caitlin Sadowski, Jaeheon Yi, Kenneth Knowles, and Cormac Flanagan

Difference between atomic and synchronize in Java?

Bizarre statement that the easy cases didn't compress well?

Mechanizing the Metatheory of a Language With Linear Resources and Context Effects

Daniel K. Lee, Derek Dreyer, and Andreas Rossberg

'One of Derek's crazy languages'

Predicate for 'linearity' (not sure if I believe it really is such), explicit contexts.

Actually, hell, this funny notion of linearity looks like exactly the same damn thing as the names. At the binding site of α, create a linear variable. At definition of α, consume it. At other mentions of it, 'useless' it.

It's not quite the same, for you're only supposed to use α *after* it's defined, right? Very close, though. Maybe at that moment you consume it and rebind an unrestricted variable? That wouldn't require any zero-ary business would it? Then why aren't **new** and **def** the same?

Oh! Talked to dklee, turns out you are allowed to mention αs early, so the encoding I have in mind is probably exactly right.

Case Study: Subject Reduction for Mini-ML with References, in Isabelle/HOL + Hybrid

Alan J. Martin

Taking Cervesato and Pfenning '96 and doing it in Isabelle/HOL. Hybrid is a thing Alberto worked on to carve out the 'syntaxy' part of the full non-HOAS function space.

Something about a 'two-level' approach? What advantage do we get from such intense indirection? Do we have effective loss of confidence in adequacy?

Mechanizing Methatheory with Nested Datatypes
Andre Hirschowitz and **Marco Maggesi**

Oh so 'nested datatypes' seems to be that thing that kaustuv told me about long ago where you talk about α terms where α is the type of variables: the constructor var would have type $\alpha \to \alpha$ term.

A synonym for 'nested' seems to be 'nonregular'.

Bird and Meertens '98 looks like the canonical paper. They make an analogy between regular type definitions and *tail-recursive* functions. Nested datatypes are just the more general case where appeals to the recursively bound variable are not necessarily tail-recursive.

An environment is a coalgebra for ty? (ty being of course the polymorphic datatype that is like term above, but he's talking about *type* expressions with variables) The same as the type of the var constructor. tymap is a coalgebra map?

Adam Chlipala asked why he didn't just use deBruijn indices with a dependent type discipline — the speaker responded with something about algebras over a monad, but I think the crux is that the nested datatype approach amounts to shoehorning the type of naturals into the type language, which makes some later expressions shorter, for lack of a need to make that translation *explicit*.

Shallow embedding of a logic in Coq
Jerome Vouillon

Didn't really understand this one.

SASyLF: An Educational Proof Assistant for Language Theory
Jonathan Aldrich, Robert J. Simmons, and Key Shin

User-friendly second-order front-end to some of Twelf's metatheorem checking facilities.

Building Verified Language Tools in Operational Type Theory **Aaron Stump**

Optimized LF type-checking — incrementally parse and typecheck, and avoid building ASTs for terms that are going to be garbage eventually anyway. Is typechecking really a bottleneck, though?

I'm naturally suspicious of so much naming...

"Tackling the Awkward Squad" nice old paper by Peyton-Jones et al.

Twelf raises exception on one of Stump's example suite? And timing out on others?

God damn the subjectivity of the size or import of the trusted code base drives me crazy still.

5-minute talks

Forgot his name: 'House' operating system mentioned. Something about separation logic and coinduction for reasoning about schedulers.

Andrew Kennedy: .NET subtyping subtle, should formalize it.

Stephanie Weirich: "On why twelf is better than Coq". She backs off to a more LISPy datastructure and carves out a refinement so that substitution is shorter to define.

`http://www.cis.upenn.edu/~baydemir/abstracting-syntax/`

Adam Chlipala says: if you had enough automatic code generation, wouldn't that be enough? Stephanie says: maybe with type classes?

Adam Chlipala's own talk: sounds like Meta-Tarskian circularity, just within the world of Coq.

I'm kind of sick of the bullshit religious wars between Coq and Twelf in particular.

Karl Crary promoting the Penn paper "Engineering Formal Metatheory". He makes the point that we need variables to α-vary not only in terms in the object language, but in the derivation. I'm kind of worried his pedagogical suggestion means a return to that awful vertical 'dot dot dot' notation for natural deduction.

The thing that bothers me about the conceptual primacy of variables that are amenable to (ordinary) substitution is that ordinary substitution is just *one* thing we like to do to bound things (and weakening and contraction are just some old properties that we *sometimes* like to hold of them) and not the only one. Hereditary substitution being another, obviously, and how are we to internalize that?

2008.9.21

From 2007.10.17, the idea that nature has an interest rate.

In point of fact, it has many; various investments may have different rates of return.

Consider the ethical objection to usury: that a rich person is being exploitative by lending money to a poor person at interest. But if they are not allowed to charge interest, they would prefer to buy equipment that *would* yield such an interest rate. So either we coerce the rich to offer interest-free loans, or we suffer with people not being able to afford houses. Possibly, however, we might prefer "external" investment because then more of 'nature' is being exploited?

2008.9.22

Lazy and Speculative Execution

Butler Lampson

Laziness is not computing (yet) something the programmer told you to compute, but which you bet won't be needed.

Speculativity is computing something the programmer didn't tell you to compute (yet), but which you bet will be needed.

They appear tantalizingly dual.

Kind of like put and call options for computation?

Something about 'escrow locking'.

The locking in concurrent version control is, looked at correctly, a kind of speculativity or laziness, being optimistic that maybe nobody will interfere with your file you checked out.

2008.9.23

Defunctionalized Interpreters for Programming Languages

Olivier Danvy

Consider the Scott-Tarski interpreter that implements the lambda calculus in terms of... the lambda calculus. If the 'host' language is CBN then the interpreter is, and respectively CBV. So the *content* of CBV and CBN is in the respective CPS transforms.

2008.9.24

Polymorphism and Page Tables–Systems Programming From a Functional Programmer's Perspective

Mark Jones

He emphasizes *separation* of processes — this is well and good, but experience using Chrome for instance shows me that this isn't enough. What I also want are minimum-resource-guarantees to each process. For using up all the damn CPU (or RAM, or disk) *is* after all a way for processes to interfere with one another.

What are the obstacles to compiling FP programs to a target language *without* GC?

2008.9.25

Still trying to work through the idea that came up during dinner the first day of ICFP proper about pattern comatches. Resembles bunched logic with a distributivity property.

Went to a talk by Larry Lessig. Inspiring but not convincing.

2008.9.26

Realization: conformal maps are locally orthogonal, a strengthening of how differential maps are locally linear. Conformal maps do *not* necessarily

preserve dot product, just dot product divided by lengths.

2008.9.27

What is the mixture of hard and easy problems in the following setup: each agent has a collection of goods, and advertises which transactions they feel desirable (in the form: "I receive X, Y, Z and give W, V, U")

Consider the essential things in this model that make it realistic. The (typical) composability of desirable trades, the properties of physical objects like non-copyability and (default) indestructibility.

Consider the role the law plays in constructing ownership. It is not that we have an intrinsic, canonically defined right to property — I can't imagine how the precise boundaries of that right would be inferrable from first principles.

Ownership falls out of the *highly contingent rules of the game* that government guarantees.

In a sense there are 'natural' rules about how it is easier to control things that are physically near to you, that you have in a locked warehouse, etc.

But the institution of governments says: let us live in a fictive world where theft is impossible, where it is not an allowable move in the game. If you break the magic circle of the game by going outside its rules, then it is empirically likely you will be punished. What is mysterious is the rationalization of the enforcers; they live on the boundary of the magic circle, or perhaps in a slightly enlarged one, where the meta-rule is, 'if you break the (non-meta-)rule, you must be punished'.

I remember the distinction between Platonic and Pragmatic rules from a discussion in (perhaps it was) Agora Nomic. Platonic rules say: 'thou shalt not'. Pragmatic rules say: 'if thou doest, then thus'. But the trouble is, *all* laws are ultimately pragmatic, or at least have a pragmatic re-reading. The stability of the system *relies* critically on mutual enforcement or complicity of all agents in the belief in the rule of law, and of one particular legal inheritance.

2008.9.28

Oh, inserting the stuff that `%hlf` does messes up line numbering.

Naming is also still fucked up. Can't remember exactly what my conjecture about that was.

Any thoughts I had about checking coverage in LF and transferring don't seem too plausible right now. Staring at the two commutative cases for the right rule of tensor in the second derivation, I feel in my guts the need for reasoning *from* labels *to* higher-order apartness.

I recall that it was higher-order worlds that I thought could conceivably be used to uniformize the fact that a linear variable could go either way.

2008.9.29

Tried to do the cps translation in HLF. Seems it chokes on application?

```
%hlf.

e : type.
v : type.
app : e -> e -> e.
lam : (v -> e) -> v.
inj : v -> e.

ce : type.
cv : type.
co : type.
capp : cv -> cv -> co -o ce.
cth : co -o cv -> ce.
cfn : (cv -> co -o ce) -> cv.
clam : (cv -> ce) -> co.

cps : v -> cv -> type.
cpse : e -> (co -o ce) -> type.

cps/lam : cps (lam E) (cfn E')
  <- ({x:v}{x':cv} cps x x' -> cpse (E x) (E' x')).
cpse/app : cpse (app E1 E2)
     ([c :^ co] E1' ^
     (clam [w1] E2' ^
     (clam [w2] capp w1 w2 ^ c)))
  <- cpse E1 E1'
  <- cpse E2 E2'.
cpse/inj : cpse (inj E) ([c :^ co] cth ^ c E')
  <- cps E E'.
```

2008.9.30

My mistake was forgetting linearity in `clam : (cv -> ce) -o co.`

Remarkable how the linear continuation arguments in here are exactly the same as monadic conclusions.

2008.10.1

Funny counterexample; yields an attempt to generalize over higher-order worlds in the last declaration when I wouldn't suspect it would want to.

```
%hlf.
```

```
tm : type.
name : type.

lam : (name -0 tm) -> tm.
var : name -0 tm.
app : tm -> tm -> tm.

diseq : tm  -> tm -> type.

#' : name -0 name -0 type.
%abbrev # = [a] [b] #' ^ a ^ b.
%infix none 5 #.

diseq/var/var : {a:w} A # B -o diseq (var ^ A) (var ^ B) @ a.

diseq-sym : {a:w} diseq E2 E1 @ a -> diseq E1 E2 @ a -> type.
apart-sym : {a:w} (N1 # N2) @ a -> (N2 # N1) @ a -> type.

diseq-sym/var/var :
     diseq-sym ^ (diseq/var/var ^ ^ APART)
                                 (diseq/var/var ^ ^ APART')
     <- apart-sym ^ APART APART'.
```

This reduces to

```
%hlf.
o : type.
# : o -0 type.
ax : (# ^ B) -> o.
r : {a:w} o @ a -> type.
c : r ^ (ax N).
```

which I can repair by doing

```
%hlf.
o : type.
# : o -0 type.
ax :  {a:w} {B: o @ a} (# ^ B) -> o.
r : {a:w} o @ a -> type.
c : r ^ (ax ^ B N).
```

The original one seems to work by explicitizing `apart-sym` like so:

```
%hlf.

tm : type.
name : type.

lam : (name -0 tm) -> tm.
var : name -0 tm.
app : tm -> tm -> tm.

diseq : tm  -> tm -> type.

#' : name -0 name -0 type.
%abbrev # = [a] [b] #' ^ a ^ b.
%infix none 5 #.

diseq/var/var : {a:w}
% {aA : w} {A : name @ aA} {aB : w} {B : name @ aB}
 A # B -o diseq (var ^ A) (var ^ B) @ a.

diseq-sym : {a:w} diseq E2 E1 @ a -> diseq E1 E2 @ a -> type.
apart-sym : {a:w} {aA : w} {A : name @ aA}
                  {aB : w} {B : name @ aB}
               (A # B) @ a -> (B # A) @ a -> type.

diseq-sym/var/var :
     diseq-sym ^ (diseq/var/var ^ ^ APART)
                                 (diseq/var/var ^ ^ APART')
     <- apart-sym ^ ^ N1 ^ N2 APART APART'.
```

With *no* help apparently offered by the commented-out explicitization of `diseq/var/var`, which I had thought was the analogue of `ax` in the reduced version. Mysterious.

Really ought to implement the 0-ary pi.

2008.10.2

If one bets on linear combinations of securities, the 'spread' in joint probability space seems like it ought to be a convex subset of it.

2008.10.3

I should be able to do unification more explicitly with substructural names, shouldn't I?

2008.10.4

Still can't figure out call-by-name CPS.

2008.10.5

Here's a typed version of call-by-value:

```
%hlf.

tp : type.
o : tp.
=> : tp -> tp -> tp. %infix right 3 =>.

e : tp -> type.
v : tp -> type.
app : e (A => B) -> e A -> e B.
lam : (v A -> e B) -> v (A => B).
inj : v A -> e A.

ce : type.
cv : tp -> type.
capp : cv (A => B) -> cv A -> (cv B -> ce) -o ce.
clam : (cv A -> (cv B -> ce) -o ce) -> cv (A => B).

cps : v A -> cv A -> type.
cpse : e A -> ((cv A -> ce) -o ce) -> type.

cps/lam : cps (lam E) (clam E')
   <- ({x:v A}{x':cv A} cps x x' -> cpse (E x) (E' x')).
cpse/app : cpse (app E1 E2)
    ([c :^ (cv A -> ce)] E1' ^ ([w1] E2' ^ ([w2]
                                        capp w1 w2 ^ c)))
    <- cpse E1 E1'
    <- cpse E2 E2'.
cpse/inj : cpse (inj E) ([c :^ (cv A -> ce)] c E')
    <- cps E E'.
```

2008.10.6

An attempt at call-by-name:

```
tp : type. %name tp B.
o : tp.
=> : tp -> tp -> tp. %infix right 3 =>.

e : tp -> type.
app : e (A => B) -> e A -> e B.
```

```
lam : (e A -> e B) -> e (A => B).

ce : type.
cv : tp -> type.
capp :
 cv (A => B) -> ((cv A -> ce) -> ce) -> (cv B -> ce) -> ce.
clam :
 (((cv A -> ce) -> ce) -> (cv B -> ce) -> ce) -> cv (A => B).

cps : e A -> ((cv A -> ce) -> ce) -> type.

cps/lam : cps (lam E) ([k] k (clam E'))
   <- ({x:e A}{x':(cv A -> ce) -> ce} cps x x' ->
                                        cps (E x) (E' x')).
cpse/app : cps (app E1 E2)
    ([k] E1' ([a] capp a E2' k))
    <- cps E1 E1'
    <- cps E2 E2'.
```

2008.10.7

Now the following in uniftwelf doesn't terminate! Based on `lp` from the examples directory.

```
o : type.
pf : o -> type.
forall : (o -> o) -> o.
forallb : ({a:o} pf (A a)) -> {T:o} pf (A T).
whr : pf A -> pf A -> type.
whr_forall : whr (forallb D T) (D T).
```

Compiling fontforge again on the Eee, I also needed to tell it where X was:
`--x-libraries=/usr/X11R6/lib --x-includes=/usr/X11R6/include`

2008.10.8

Remembering and reconstructing some stuff about weak ω-categories:

$$\frac{\alpha' : c^n f \equiv c^n g \qquad \alpha : d^n g \equiv d^n f \qquad id^{n-1}_{\alpha'} \circ_{-n} f \circ_{-n} id^{n-1}_{\alpha} \sim_n g}{f \sim_{n+1} g}$$

$f \sim_{n+1} g$ reasonable to ask if $d^n_* f = d^n_* g$.

2008.10.9

Need to nail down a good example of coverage checking on worlds specifically.

2008.10.10

Should check if things work with double-check turned on after I fix this termination bug.

2008.10.11

A 'topologization' of a category should be something like subdividing it until you 'can't tell the difference'. Does topological continuity fall out of Scott-style (and ultimately Brouwer-style) *computational* continuity?

2008.10.12

Could hypersequents be generalized to linear logic? To a more general notion than disjunction at the 'hyper' level?

Have been poking around with SMLNJ's FFI features. They're pretty under-documented.

2008.10.13

Finally got a feel for what I think Girard has been getting at about coinductive proofs of 'sanity' for, say, linear logic. In the classical setup logical connectives are nearly just arbitrary operations on derivations, but *it so happens* that they come in dual pairs which satisfy identity and cut elimination.

2008.10.14

Peter Lumsdaine's talk was interesting again today. He said he didn't see anything obviously wrong with my approach to weak ω-categories. I still have relatively little confidence as it is.

2008.10.15

Started just writing code in SML to speak the X11 protocol directly.

2008.10.16

Working pretty well. I can get interval timers nicely by just passing arguments to `select`.

2008.10.17

Had a nice set of ideas today, starting during lunch.

The first intuition was, okay, how do we get a good, axiomatic notion of topology with direction? (apart from existing attempts based on slapping a local poset structure on a space) I tried to imagine a closure operation

that acted asymetrically, which took only limit points that were approached 'from one direction'. What axioms might that satisfy? In fact, all the usual ones! I thought for a moment I didn't even need to step outside the category Top to get the theory I wanted. After all, the upper limit topologies on $\mathbb{R}$ are perfectly sensible topologies.

But then eventually I remembered that these topologies lack all kinds of nice properties — they make the reals totally disconnected, for one. I thought about *imposing* ordinary continuity on top of, say, upper limit continuity, but that seemed horribly asymmetric.

Then I convinced myself that a function that is both upper limit continuous and lower limit continuous iff it is continuous and monotone. So the solution seemed to be to require maps to be at once continuous in both topologies.

2008.10.18

Whittled down Anders's twelf coverage counterexample a bit.

```
o : type.
%block b : block {w:o}.

a : o -> type.

pred : (a X' -> a X) -> type.
pred/case : pred D.

rel :  a X' -> a X -> type.
rel/case : rel D D.

thm : ({h:a X} rel (D1 h) (D2 h))
    -> pred D1 -> pred D2 -> type.
%mode thm +R +P1 +P2.

%worlds (b) (thm _ _ _).
%covers thm +R +P1 +P2.
```

This yields a Bind exception.

2008.10.19

Read some stuff about model categories.

On the one hand, there seems to need to be work done to get local pospaces to even be a cocomplete category, whereas multiple topologies seem to obviously have colimits.

In fact I bet the reason they do has to do simply with taking limits in Cat of diagrams of categories and functors that appropriately preserve

(or reflect, or create, maybe) colimits. For the category of sets with two topologies is just the pullback of the forgetful functor from Top, isn't it?

On the other hand, the whole enterprise of model categories seems to be to avoid size issues when trying to localize — i.e. create formal inverses for — a certain collection of morphisms. From some kind of topological standpoint, all that this is doing is *removing* structure, i.e. destroying open sets. I wonder if this programme could be pushed through and sidestep all these fiddly size issues?

Too bad I don't know what the homotopy theorists want to quotient out by weak equivalence *for*, exactly. It's certainly not that I can't *imagine* it being desirable, I just don't know a specific theorem that I could prove by better means.

2008.10.20

Realized the sort of obvious fact that the Sierpinski space is dually the 'coshape' of the 'cocells' in a topology that are open sets, which you discover by maps into it from a generic category, just like the one-object and one-arrow categories are the shapes of cells in a category, discovered by maps out of them.

It's very obvious that nTop is not embeddable in Top. You can classify objects by the number of points they have, and there are different numbers of two-point objects in each nTop ($2^{n-1}(2^n + 1)$).

I wonder about Chu spaces, though.

2008.10.21

Peter Lumsdaine's talk actually defined weak ω-categories a la Leinster today: they are algebras for the initial globular-operad-with-contraction. The contraction machinery is beautifully mysterious. Just by demanding one-directional connectnedness of any two ways of composing the same data, you seem to get everything you need.

2008.10.22

Okay, at least a patch to unification from last month doesn't have the termination bug.

2008.10.23

Another obvious notion of co-cell in topology: maps to the two-point discrete topology $((*)(*))$ are 'disconnections' of a space.

Is this all cohomology is?

The higher-dimensional generalizations of this seem like they ought to be maps into the boundaries of spheres.

To map B_{n+1} into a space in such a way that two points are both in the image shows that they are connected; to map a space into S_n in such a

way that two points are separated shows that they are disconnected.

I'm tempted to try to show: if there is no 2-cell from f to g inside some space X, then in fact there is a map from X to the 'oriented circle' that has two segments running from left to right. But this is very much like path-disconnectedness implying disconnectedness, which does not generally hold.

Take the extended long line; we lack evidence that it is disconnected (because we cannot form a map to 2) and evidence that it is path-connected (because it is too long to be covered by a path) This seems very intuitionistic.

Let $b : B \to X$. We say b is h-path-connected (for $h : B \to C$) if there is a path $\pi : C \to X$ such that $\pi h = b$. We say b is disconnected if there is a retraction r such that $rb = \mathsf{id}$.

Easy fact: if b is h-path-connected and disconnected, then h is disconnected.

Double topologies *nearly* embed into the category Bubenik uses in one talk, where you take a topology together with a class of directed paths that are supposed to be closed under composition with each other and also monotone nondecreasing maps from [0,1] to itself.

A counterexample to the obvious embedding you'd try is again the (this time directed) extended long line. Depending on whether we make the extended 'point at ω_1' open or not, it is disconnected or connected. Yet the obvious embedding into the category of topologies-with paths doesn't detect this potential connection. The result is path-disconnected in either event.

No, wait, I stil detect a difference in that the overall topology detects the connection. Still, I'd conjecture the obvious embedding is not injective on isomorphism classes of objects.

2008.10.24

The fact that composition works at all in a category seems to be a kind of Siefert-van Kampen theorem... The coequalizer in Cat of two arrows joined end-to-end has only one essential arrow from beginning to end. Conceivably it has more than one, but they had better be glued together universally, i.e. contractibly.

It seems to be the case that the interval **2** is then appropriately a weak coequalizer.

Luis Caires told me a bit about session types; they seem to have to do

something with linear cut elimination. Reads and writes are dual 'pure' indexed abstract modalities, I think.

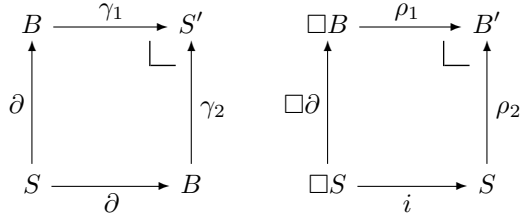
2008.10.25

I have been thinking lately about the tower of spheres S_n and balls B_n.

Imagine we have a functor $\Box : \mathbb{C} \to \mathbb{C}$ and natural transformations $c, d : \mathsf{id} \to \Box$ and $i : \Box \to \mathsf{id}$. Think of $\Box$ as something like $(\mathbf{2} \times —)$. it 'cubicalizes' an existing object. We require $id = ic = \mathsf{id}$, and call such a functor a *boxad*. A morphism of boxads $\Box_1 \to \Box_2$ is a natural transformation $\alpha : \Box_1 \to \Box_2$ such that $\alpha c_1 = c_2$, $\alpha d_1 = d_2$, and $i_2 \alpha = i_1$, as you'd expect.

Given a map $\partial : S \to B$ (which we imagine to be a coboundary map from an $(n-1)$-dimensional sphere to an n-dimensional ball) we can come up with the sphere and ball of the next higher dimensions.

Compute the two pushouts



The appropriate map $\partial' : S' \to B'$ results from using the left pushout's universal mapping property on $\rho_1 c$ and $\rho_1 d$. It follows from naturality of d, c that the pushout UMP is applicable. There's a bunch of other maps we can grab (such as $I : B' \to B$ from the UMP of the right pushout on $i_B x$ and ∂) that mean we have actually defined two boxads in the coslice category $S/\mathbb{C}$. The arrow across the left pushout $S \to S'$ is the result of applying one of them, and the arrow $\rho_2 : S \to B'$ is the other. And of course ∂' is a map of boxads from the one to the other.

But in general if a category has coproducts, then $(X + X, \mathbf{inl}, \mathbf{inr}, [\mathsf{id}, \mathsf{id}])$ is bound to be the initial boxad, and since we tacitly assumed pushouts, the coslice category has coproducts.

So it's no surprise there's a good map $S' \to B'$! For S' arises from the initial boxad applied to ∂. The curious thing is how we lifted the boxad $\Box$ to a boxad on the slice category.

If we iterate this construction, we get a nice 'globular object' B_n with codomain and domain maps going up. The globularity condition falls right out of commutativity of pushout squares.

The thing is, since the spheres always come out of the initial boxad, perhaps the balls should come from a final boxad? If its finality is somewhat weak — if it's something that smells like 'up to homotopy' — then this

actually seems rather reasonable in Top. Then making a sphere out of something is the free way of making it contractible.

I really would like to say something like: 'a collection of things like weak ω-categories' is a category with a (weakly) initial and (weakly) final boxad, maybe satisfying some further axioms. One probes the 'cells' of an object X by studying maps from B_n into X. A homomorphism $X \to Y$ is just a map in the category. A 2-cell between f and g is a map $\alpha : \Box X \to Y$ such that $\alpha d = f$ and $\alpha c = g$. 'And so on,' though I'm not quite sure what that means.

I think I need some changes in policy.

One: paradoxical anti-procrastinatory policy of no work after 5pm, but simultaneously requiring that I get some work done every day.

Two: break T category in work requirement system into more subcategories so that no project among those that require continuous effort gets left behind.

(**TW**) Thesis writing

(**TFU**) Required programming: unification

(**TFC**) Required programming: coverage

(**TE**) Required email or other communication, if any outstanding

Just now did

(**TW**) Copied in some of the section on substructural nominal logic, worked on introduction

(**TFU**) Poked at some debugging code on Anders's counterexample

(**TFC**) Ran kolm on svntwelf-20080917; broke as before

(**TE**) Emailed Brigitte back

2008.10.26

Game idea: something like bridge bidding with OSPD words. An incentive to repeat words. An incentive to have your convention be partially discoverable even by your 'oppponents'.

(**TW**) Cleaned up the section on substructural nominal logic

(**TFU**) Wrote better debugging code, made progress understanding the role of LVars.

(**TFC**) Wrote the following code to run regression tests. Ran them all with doubleCheck turned on, and they still work, with the exception of course of `kolm`.

(**TE**) Work-related email queue empty. Paid the electric bill.

```
fun dir "XXX" = raise Match
  | dir s = ("/home/jcreed/sb/twelfs/svntwelf/examples/"
             ^ s ^ "/")

fun cfg "XXX" = raise Match
  | cfg _ = "sources.cfg"

fun runtest s =
    let
val _ = print ("=== Running test " ^ s ^ "\n")
    in
OS.FileSys.chDir (dir s);
Twelf.make (cfg s)
    end

fun go () =
       let
       in
   runtest "arith";
   runtest "ccc";
   runtest "church-rosser";
   runtest "compile/cls";
   runtest "compile/cpm";
   runtest "compile/cps";
   runtest "compile/cxm";
   runtest "compile/debruijn";
   runtest "compile/debruijn1";
   runtest "cpsocc";
   runtest "cut-elim";
   runtest "fj";
   runtest "fol";
   runtest "guide";
   runtest "handbook";
   runtest "incll";
   runtest "js4";
(*    runtest "kolm"; *)
   runtest "lp-horn";
```

```
   runtest "lp";
   runtest "mini-ml";
(*     runtest "modal"; *)
   runtest "polylam";
   runtest "prop-calc";
(*    runtest "small-step"; *)
(*    runtest "tabled"; *)
   runtest "tapl-ch13";
   ()
       end
```

2008.10.27

(**TW**) Took out a section reference that doesn't exist. Am I even going to mention much about ordered logic programs in HLF beyond pure speculation?

(**TFU**) Imported debugging code from the debug.svntwelf branch.

(**TFC**) Found a fix that works, but my understanding of the secod substitution in an LVar is still sketchy. Emailed Frank asking what was up with it.

(**TE**) Paid gas bill. Emailed gwillen and said I would call Paula about the age of the house.

2008.10.28

My notion of 'boxad' is called a *cylinder endofunctor* in Grandis's writing.

Lots of good stuff in Applied Categorical Structures 15(4).

(**TW**) Removed some more bad section references. Need to look up where various lemmas are coming from.

(**TFU**) Took another look at the discrepancy between kolm under the new and old code. Seems like very small potatoes, and every evar in sight seems to support the amount of eta-expansion happening.

(**TFC**) Removed printfs so I could run my code on anders original example. It gives a coverage error. I suppose this to be correct? I am not sure.

(**TE**) Emailed rob simmons back about talt regression. Balanced checkbook.

2008.10.29

Advisor meeting today; need to rethink coverage for blocks.

2008.10.30

(**TW**) I think I copied and pasted proof irrelevance junk in here for some reason.

(**TFU**) Safety of eta-expansion is asymmetric, remember; it's 'okay' to contract, but not to expand. I wonder if aggressively contracting instantiations might work?

(**TFC**) Block declarations are like indexed type definitions. Can I squeeze this intuition to any purpose?

(**TE**) Need to call health insurance provider.

2008.10.31

(**TW**) Made a couple more small edits.

(**TFU**) Tried some more eta-collapsing. Didn't help yet.

(**TFC**) Ran new unification code on talt suite. Seems to work. Is talt the one I want?

(**TE**) Called insurance people. Apparently I'm liable for the first $350 per year. That's pretty fucking annoying.

2008.11.1

(**TW**) Thought about linear token monad.

(**TFU**) Tried a different spot, still didn't work.

(**TFC**) No, I wanted ts-lf. Breaks with double-check on.

(**TE**) Mailed check for allergist bill.

2008.11.2

(**TW**) Fixed part of statement of substitution theorem

(**TFU**) Tried to eta-collapse after invertSub in *recursive* position (when called by `invertEVarW'`) rather than in the interface, and this seemed to work, and even fixed the bug. This merits further investigation, but is very exciting.

(**TFC**) Shrank rob's counterexample.

(**TE**) Queue empty

Actual (say, catalysis-centric) chemical reactions are poorly modelled in linear logic because of a lack of negation to represent quiescent self-transitions — but it seems one can use non-exponential modalities to make up for it.

2008.11.3

(**TW**) Abstract

(**TFU**) Diffed the outputs — seem to be just the same!

(**TFC**) Rob's bug seems to have gone Heisen-. At chatter:=7, the bug vanishes. Something about the printExp in finitary1 in cover.fun

(**TE**) Need to talk to Jen about reimbursements, Ryan about deposits

2008.11.4

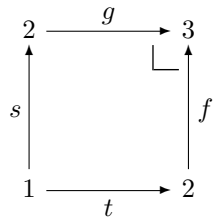
(**TW**) Acknowledgments

(**TFU**) Ran old tests. Seems to work!

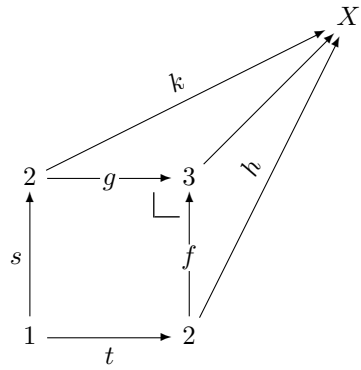
(**TFC**) The heisenbug stems from whnf doing lowering imperatively. Frightening! Maybe it should be trailed is all, though? The ultimate cause once I stop that, though, is something in the coverage of lvars again I think.

(**TE**) Voted.

Suppose I try to axiomatize the category of ω-categories. Certainly there are objects 1, 2, and arrows $s, t : 1 \to 2$. I also know there to be an object 3 and arrows $f, g, gf : 2 \to 3$.



I expect this to be a kind of weak pushout, so that if I have an 'equality' between ks and ht for $h, k : 2 \to X$, i.e. a 2-cell, I get a suitably universal 1-cell.



2008.11.5

(**TW**) Edits

(**TFU**) Even with the fix, the full kolm doesn't work, trying to abstract over a substitution that has an underscore still left in it apparently.

(**TFC**) Tried to trail printing, but CSManager doesn't seem to exist there. Not sure what to do.

(**TE**) Paid rent and water bill. Need to note how much it is.

2008.11.6

Here is an improved idea for defining a weak ω-category.

Abbreviate strict ω-category by soc. Likewise 'woc'.

An soc is *object-contractible* if for any two objects A, B in it, there exists an arrow $f : A \to B$. It is said to be *contractible* if it and all its hom-socs are object-contractible.

Two objects in an soc are *contractibly equivalent* or just *equivalent* if there is a subsoc containing both of them.

A woc is a subset W of the cells of an soc S (called a woc 'based on S') that is 'contractibly full' i.e. for every object in S there is an equivalent object in W, and likewise for every pair of objects A, B in W there is a woc based on $S(A, B)$.

I was tempted to say that a map between wocs is a map between socs that preserves the subset. This doesn't seem to work, however. Take the soc 3 that has two arrows that compose to a third, which is also a woc by including everything. Take also the soc $3'$ that has another arrow in the same niche as the composite, which is equivalent to it, and the woc that includes everything.

I feel that I should be able to have a weak map $3 \to 3'$ that sends the composite not to the composite, but to its equivalent doppelgänger. The requirement of preserving the subset together with the requirement that ω-functors preserve strict composition prevents this.

So maybe a woc-map is just a soc-map, and we shift our expectations of what a functor tells us. Instead of giving *the* object (arrow, 2-cell, ...) that is the result of hitting an object (arrow, 2-cell,...) with the functor, we only find out *which* cells are valid outputs.

(**TW**) Edits

(**TFU**) Futzed with the remaining bug in kolm. Actually getting a case where I have unresolved constraints where before I didn't.

(**TFC**) The Inst branch of the type Block is used in tomega. My version of the counterexample from rob seemed broken. Fixed it. Spent a while writing a little bit of trailing code for whnf, can't confirm that it is actually useful, though.

(**TE**) Paid electric bill.

2008.11.7

(**TW**) Edits.

(**TFU**) Generated a slightly smaller counterexample for kolm.

(**TFC**) Poked at coverage code — still a surprising typing error showing up.

(**TE**) Emailed ryan about deposit.

Talked to neel about logical relations and stuff.

It turns out the stuff at the heart of it is quite simple.

Suppose I have types A and B and a binary relation R over values of A, B. It is possible to lift R to a binary relation $C(R) : C(A) \to C(B)$ for any type operator $C(\alpha)$. At α I just get R, at base types I take the identity relation, and at types derived from the cartesian closed structure of types, I use the cartesian closed structure of relations.

The fundamental of logical relations is just that for any term M : $\forall\alpha.C(\alpha)$, we get $(M[A])C(R)(M[B])$.

How can I generalize this? To go from types to pairs of types I have some functor $D : \mathbf{Cat} \to \mathbf{Cat}$. Then from a pair of types I should be able to get a notion of relation, i.e. a category $\mathbf{R}$. So this is like an indexed category sitting over $D(\mathbf{C})$, as a functor $\mathbf{R} \to D(\mathbf{C})$. Remember, $\mathbf{C}$ is just the type theory we started with. So now C is a type operator, a map $\mathbf{C} \to \mathbf{C}$. Don't know how to go any further.

2008.11.8

Is it the case that we can still analyze the canonical forms of e.g. $\forall\alpha.(\alpha \to \alpha \to \alpha) \to ((\alpha \to \alpha) \to \alpha) \to \alpha$?

2008.11.9

(**TW**) Edits.

(**TFU**) On further counterexamples for kolm: compare against clean code to make sure intermediate stages are actually things I *expect* to reasonably construct.

(**TFC**) The code of Karl's that frank Forwarded me seems to be ts-lf, with the same bug! Except on my code I get the double-check violation, and on clean code I get a hang.

(**TE**) Checked about meeting tomorrow.

A positive Σ type almost begins to make sense if asynchronous decomposition of positive types (i.e. pattern decomposition) takes place in the *opposite* order that I am accustomed to — that is, if you decompose types *earlier* in the dependency order first.

Alternatively you can make Σ sort of a $\otimes$!-like thing, where it creates one pattern variable and one ordinary variable (the ordinary one being

the bound one) and then the order of asynchronous positive decomposition doesn't matter.

2008.11.10

I would claim that block declarations can be reduced without loss of generality to the case of 'some Γ_s block a Γ_s'. This doesn't totally settle the question of which substitutions need to be tracked on world variables, but it does shed some light on it.

2008.11.11

Base-type polymorphism can still support 'lists of higher-order functions' to some extent. One needs only inject (say) $nat \to nat \to$ type into some other base type.

Grandis puts aside bitopologies for higher-dimensional algebra on the basis that you don't get an appropriate adjoint to $\times\mathbf{2}$. But I suspect that isn't the right cylindrification functor at all! Rather one that takes advantage of an infinite family of topological structures on the carrier — the Gray product looks very promising here.

Coverage checking in regular worlds asks the following question. Given a context Γ that satisfies a given regular-world assumption, and a type A over $u_1, \ldots, u_n$, do there exist any terms $M_1, \ldots, M_n$ coming from that Γ that, when substituted for $u_1, \ldots, u_n$ yield a type that has inhabitants in Γ?

Here's my best counterexample to completeness of new unification so far:

```
i : type.  %name i T.
o : type.  %name o A.
not    : o -> o.        %prefix 12 not.
exists : (i -> o) -> o.
nk : o -> type.
nk_noti    : ({p:o} nk A -> nk p) -> nk (not A).
nk_note    : nk (not A) -> {C:o} nk A -> nk C.
nk_existsi : {T:i} nk (A T) -> nk (exists A).
nk_existse : nk (exists A) -> ({a:i} nk (A a) -> nk C) -> nk C.

nk_dnotr   : nk A
     <- nk (not (not A)).  % double negation version of excluded middle

nk_dnotx = ([NK] (nk_noti [p:o] [u:nk (not A)] (nk_note u p NK))).

n = [p:o] (not not p).

kolm : o -> o -> type.

kolm_exists : kolm (exists A) (n (exists A*))
    <- ({a:i} kolm (A a) (A* a)).

equiv : kolm A A* -> (nk A -> nk A*) -> (nk A* -> nk A) -> type.

equiv_exists : equiv (kolm_exists K)
                     ([v:nk (exists A)]
                        (nk_dnotx
```

```
                        (nk_existse v
                           ([a] [u:nk (A a)]
                              (nk_existsi a (NK* a u))))))
                     ([v:nk (n (exists A*))]
                        (nk_existse (nk_dnotr v)
                           ([a] [u:nk (A* a)]
                             (nk_existsi a (NK a u)))))
   <- ({a} equiv (K a) (NK* a) (NK a)).
```

Also this version of it generates a Match in abstraction

```
o : type.  %name o A.

exists : o.
nk : o -> type.
nk_noti    : ({p:o} nk A -> nk p) -> nk ( A).
nk_note    : nk ( A) -> {C:o} nk A -> nk C.
nk_existsi : {T:o} nk (A T) -> nk (exists ).
nk_existse : nk (exists ) -> ({a:o} nk (A a) -> nk C) -> nk C.

nk_dnotr   : nk A
     <- nk ( ( A)).  % double negation version of excluded middle

nk_dnotx = ([NK] (nk_noti [p:o] [u:nk ( A)] (nk_note u p NK))).


kolm : o -> o -> type.

kolm_exists : kolm (exists ) ((exists ))
    <- ({a:o} kolm (A a) (A* a)).

equiv : kolm A A* -> (nk A -> nk A*) -> type.

equiv_exists : equiv (kolm_exists K)
                     ([v:nk (exists )]
                        (nk_dnotx
                          (nk_existse v
                             ([a] [u:nk (A a)]
                                (nk_existsi a (NK* a u))))))

   <- ({a} equiv (K a) (NK* a) ).
```

(**TW**) Edits.

(**TFU**) Discussion of kolm bug above.

(**TFC**) Discussion of coverage above.

(**TE**) Cashed Andreas's check.

2008.11.12

So to simplify blocks down to a special case, we change `%block b : some` Γ_s `block` Δ to

ℓ : $\Gamma_s \to$ `type.`
$\texttt{out}_\ell$: $\Pi\tau{:}\Gamma_s.\ell[\tau] \to \Delta[\tau]$.

which means the effective grammar is something like

$$\begin{array}{rcl} \text{Projection Heads } \beta & ::= & \texttt{out\_ \_ } b \mid \texttt{out}_\ell\ [\tau]\ L[\uparrow^k] \\ \text{Contexts } \Gamma & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, b : \ell[\tau] \\ \text{Heads } H & ::= & c \mid x \mid \beta.n \end{array}$$

$$\frac{L :: \cdot \vdash \ell[\tau] \qquad \Gamma \vdash \uparrow^k : \cdot \qquad \cdot \vdash \tau : \Gamma_s}{\Gamma \vdash \texttt{out}_\ell\ [\tau \circ \uparrow^k]\ L[\uparrow^k] : \Delta[\tau]}$$

$$\frac{\Gamma \vdash b : \ell[\tau]}{\Gamma \vdash \texttt{out\_ \_ } b : \Delta[\tau]}$$

Sigbovik poster idea: Book guy style
Pastiche of other famous posters?
Sigbovik paper: choose-your-own-adventure.

2008.11.13

The method is: write down the sentence you think you mean to say, and revise it towards greater clarity.

I retain a cautious fondness for the idea of Eprime, and for the sapir-whorfery it stands for. If we cannot change our (verbal) thoughts by changing which words we use, then what hope do we have?

Careful teasing-apart of self-loops in multitopology seems like it automatically forbids infinite pasting diagrams. Some use of filter-like pretopologies seems to still be required to require objects to map to objects and arrows to arrows, and to prevent 'thick' cells of lower dimension.

2008.11.14

(**TW**) Edits. My next main goal should be a chapter that just sets out HLF and does all its metatheory.

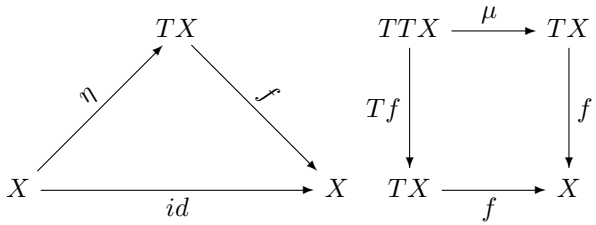
(**TFU**) More tracing of kolm bug. Now it loops, apparently?

(**TFC**) Examined uses of LVars in cover.fun. Must remember: rob's bug only happens if double-check is on.

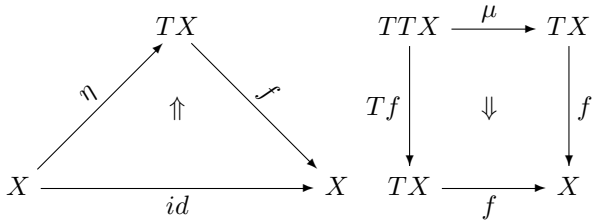
(**TE**) Balanced checkbook.

2008.11.15

Remember that old stuff about the use of cartesian monads to get at multicategories. First of all, if I have a monad T, then an algebra for T is a map $f : TX \to X$ such that



which I might want to weaken to



Secondly, if T is cartesian (preserves pullbacks and μ and η are somehow related to a pullback?) then I should be able to lift it to a monad on the category of spans. Algebras for *that* monad are consequence relations satisfying identity and cut.

Now does the two-list monad latent in Noam's focusing calculus arise from two list monads and a distributive law?

Here's another kolm plateau:

```
o : type.  %name o A.
k : o.

exists : (o -> o) -> o.
nk : o -> type.
z : nk W.
nk_noti    : ({p:o} nk A -> nk p) -> nk ( A).
nk_note    : nk ( A) -> {C:o} nk A -> nk C.
nk_existsi :  nk (B k) -> nk (exists B).
```

```
nk_dnotx : {A:o} nk A -> nk A
   = [A:o] [NK:nk A] nk_noti ([p:o] [u:nk A] nk_note u p NK).

kolm : o -> o -> type.

kolm_exists : kolm (exists A) ( (exists A*))
    <- ({a:o} kolm (A a) (A* a)).

equiv : kolm A A* -> (nk A -> nk A*) ->  type.

equiv_exists : equiv (kolm_exists K)
                     ([v]
                        (nk_dnotx _

                                    (nk_existsi  (NK*  z))))
   <- ({a} equiv (K a) (NK* ) ).
```

and this is about as small as I can make it without triggering nontermination:

```
o : type.
k : o.

e : (o -> o) -> o.
a : o -> type.
z : a A.

/ni : (a A -> a k) -> a A.
/ne :  a A -> a A  -> a C.
/ei :  a (B k) -> a (e B).
/e  : a (e A) <- ({x} a (A x)).

thm : a B -> (a A -> a B) ->  type.

thm/ : thm (/e K)
        ([v] /ni ([u] /ne u (/ei (A z))))
        <- ({x} thm (K x) ([v] A v)).
```

(**TW**) Edits.

(**TFU**) More tracing of kolm bug as above.

(**TFC**) Poked at rjsimmon-counter. Even if I abstract back the result of instevarsskip it looks right.

(**TE**) Edited accounting code.

2008.11.16

I thought that inductive types to the left of arrows might work in an LF-like setting — then I thought that they didn't because they didn't satisfy η-expansion.

That is, I can't seem to do

$$\frac{}{x : \mathbb{N} \to a \vdash (\mathbf{fold}_{\mathbb{N}>a}(z \mapsto x\ z \mid s(w) \mapsto ???) : \mathbb{N} \to a}$$

given

$$\frac{\Gamma \vdash e_1 : a \qquad \Gamma, w : a \vdash e_2 : a}{\Gamma \vdash \mathbf{fold}_{\mathbb{N}>a}(z \mapsto e_1 \mid s(w) \mapsto e_2) : \mathbb{N} \to a}$$

I believe this is because I shouldn't actually stop at one unfolding of $\mathbb{N}$ when continuing to do focusing — it really requires the ω-rule. But maybe I could deal with the type $\mathbb{N}' = \mu\alpha.1 + \downarrow\uparrow\alpha$? That on the left side of an arrow means $\mathbb{N}' \to a = (1 + \downarrow\uparrow\mathbb{N}') \to a = a \times (\downarrow\uparrow\mathbb{N}' \to a)$.

Yet this doesn't seem to help because

$$\frac{w : \uparrow\mathbb{N}' \vdash x\ (s\ w) : a???}{x : \mathbb{N}' \to a \vdash (\mathbf{fold}_{\mathbb{N}'>a}(z \mapsto x\ z \mid s(w) \mapsto x(s\ w)) : \mathbb{N}' \to a}$$

still seems to violate focusing discipline. The identity proof of $1 + \downarrow\uparrow(1 + \downarrow\uparrow(\cdots))$ is still infinite.

The focusing proof is merely infinitely tall, though, not infinitely wide, so maybe I should be able to do it in the coinductive type of proofterms. That type scares me though.

Perhaps one could think about an intuitionistically 'open-ended' set of natural-number like things that might stop, and prove theorems for any such world? That might require a partial successor function, which is weird.

(**TW**) Edits.

(**TFU**) Reduced kolm a slight bit more.

(**TFC**) The standard printing code does not distinguish between different LVars and there are even comments to this effect. Major headdesk.

(**TE**) Edited accounting code.

```
o : type.
k : o.
```

```
e : (o -> o) -> o.
a : o -> type.
z : {A:o} a A.

/ni : {A:o} (a A -> o) -> a A.
/ne : {A:o} a A -> a A -> o.
/ei : {A:o -> o} a (A k) -> a (e A).
/e  : {A:o -> o} a (e A) <- ({x} a (A x)).

thm : {A:o} a B -> (a A -> a B) ->  type.

thm/ : thm Y (/e _ K)
([v] /ni _ ([u] /ne _ u (/ei _ (A (z X)))))
<- ({x} thm X (K x) ([v] A v)).
```

2008.11.17

In kolm, I am coming up against an example where a variable is unified against itself with a pattern substitution on one side, but not the other. This is not generally safe, for I might consider the counterexample

$$u[a.b] = u[\_.\lambda y.b\ a]$$

which admits $u \leftarrow a.b.b\ a$. Also notice this works if we replace that underscore by anything else, e.g. b, which appears to conflict with a. Non-pattern substitutions are quite powerful!

(**TW**) Edits.

(**TFU**) Kolm note above.

(**TFC**) Drilled down a little further. Even the very last printf before splitting occurs is type-correct.

(**TE**) Edited accounting code. Ought to procure receipts and have them ready in case I see Jen.

2008.11.18

(**TW**) Split things up.

(**TFU**) Traced through typechecking kolm counter on the whiteboard. Didn't help much. At least I can see that no dirty tricks are required to get the unification problem solved. Maybe I just need to be more 'eager' about pruning or something?

(**TFC**) Reduced rjsimmon-counter some more.

(**TE**) Mentioned to Lea debt outstanding.

2008.11.19

Had an advisor meeting. Thought a little about higher-dimensional adjunctions.

2008.11.20

The β-reduction and η-expansion modifications seem to run opposite directions depending on whether you're dealing with a positive or negative connective.

2008.11.21

Poincaré dual diagrams should just be locally like admissible cells, and cells themselves are likewise locally 'pasting' except at their 'center point', which could be higher-dimensional than just a point.

Ruby code fed into a parametric plotter over the range $(-1, 1)$ for both x, y, to generate a picture of an adjoint unit.

```
u = x * 5; v = y * 5; t = u * u * u / 3 + u * v;
 t = t * 0.1; t > 1 ? 1 : t < -1 ? -1 : t
```

2008.11.22

(**TW**) Reboot.

(**TFU**) Reboot.

(**TFC**) Reboot.

(**TE**) Need to pay gas bill.

2008.11.23

(**TW**) Edits.

(**TFU**) Coverage needs to slacken at typing of evar expressions, not at the typing of the evars themselves.

(**TFC**) Traced deeper in coverage code. Caused a nontermination problem when I stupidly pulled a call to abstraction out of a callback.

(**TE**) Printed receipts for travel.

2008.11.24

(**TW**) Edits.

(**TFU**) Is affine slackening only appropriate for linear logic? Does general HLF need something else?

(**TFC**) Traced coverage code. The problem is in world cases. I would have thought this would mean "new worlds" but apparently it counts already existing ones as well, or perhaps even exclusively.

(**TE**) Paid gas bill.

2008.11.25

(**TW**) Edits.

(**TFU**) Could Anders' nonproofirrelevant strategy work? It seems like a big change this late. Should see what the implementation does tomorrow morning.

(**TFC**) Oh, I see Carsten already has reported a bug here.

(**TE**) Talked to Jen — need an actual receipt from the hotel.

2008.11.26

I am copying the technique for encoding arbitrary unificataion problems here from my livejournal entry of april 28 2007 because I keep wanting to find it and looking here.

```
o : type.
f : o -> o.
eq : (o -> o -> o) -> (o -> o -> o) -> type.
refl : eq M M.
c :     eq ([a] [b] X a) ([a] [b] f (Y (Z b)))
     -> eq ([a] [b] Y a) ([a] [b] X (Z b))
     -> type.
test : c refl refl.
```

2008.11.27

Say a point $v \in \mathbb{R}^n$ is 0-regular if $f(v) \neq 0$, and 0-singular otherwise. Say it's 1-regular if it's 0-singular and $f_x(v) \neq 0$, and 1-singular if it's 0-singular and $f_x(v) = 0$.

I *think* I want to say of the 1-singular points that those that have f_{xx} and f_y both nonzero are regular, and the rest are singular, but I'm having difficulty determining why this should be the right choice.

Where I get started is the observation that $f_x \neq 0$ while $f = 0$, then there's a neighborhood such that there's exactly one 0-singular point in each 1-dimensional slice of it, and all the 0-dimensional things on the left of it are related to each other in a way without 0-singularities.

For take a neighborhood where (say, wlog) $f_x > 0$ throughout. Near the centers of its x-extremities, (which have f-values below and above zero) find by continuity neighborhoods entirely below and above f. Intersect these along every dimension other than x, and consider the product of that with the original x-interval. This is the required neighborhood.

Certainly if $f_x = 0$, then we risk having more than one 0-singularity in an arbitrary nearby one-dimensional slice; just consider the graph of $y = x^2$.

Okay, now pop up to dimension two. We want to divide 1-singularities into 2-regular and 2-singular points, and classify the 2-regular points by domain and codomain.

I conjecture that 2-regular points have both $f_{xx} \neq 0$ and $f_y \neq 0$. For suppose both are positive. Find a neighborhood where they stay so. Then f_x is negative to the left and positive to the right, and f (along $y = 0$) is positive everywhere except the origin, where it's zero.

Also, we know there's exactly one point in every slice where $f_x = 0$. We can think of this as a function of y and all the other variables.

Here I get stuck.

2008.11.28

(**TW**) Edits.

(**TFU**) Reboot.

(**TFC**) Reboot.

(**TE**) Still have gas bill, need to get receipts.

2008.11.29

The first four dimensions of adjoint equivalence cells:

Simplified

(1)

$$(x, 1)$$

$$(x^2, 1, x)(x, y, 1)$$

$$(x^3, 1, x, x^2)(x^2, y, x, 1)(x^2, 1, xz, x)(x^2, y, 1, x)(x, y^2, 1, y)(x, y, z, 1)$$

Standard

$$(x)$$

$$(x^2, y)$$

$$(x^3, y, xz)(x^2, y^2, z)$$

$$(x^4, y, xz, x^2w)(x^3, y^2, xz, w)(x^3, y, xz^2, xw)(x^3, y^2, z, xw)(x^2, y^3, z, yw)(x^2, y^2, z^2, w)$$

For all disjunctions in uses of the projection lemma, (parens) indicate the one that's true by assumption, and [brackets] indicate one that's true by virtue of forbidden polynomials.

The two-dimensional case:

Suppose I encounter f^+_{xx}, f^+_y and f^0_x. I can project f_x to f along y and still get f^+_{xx} if f^0_{xy} or (f^0_x). Then I find that there's a unique singular point with f^0_x, the one I'm sitting on.

The three-dimensional cases:

Suppose I encounter $f^+_{xxx}, f^+_y, f^+_{xz}$ and f^0_x, f^0_{xx}.

I can project f_{xx} and f_x to f along y and keep f^+_{xxx}, f^+_{xz} if (f^0_x) or f^0_{xxy} and if f_z or $[f_{xy}]$. Now I appeal to the induction hypothesis thinking about f_{xx} to see that no other singular points occur too close.

Suppose I encounter $f^+_{xx}, f^+_{yy}, f^+_z$ and f^0_x, f^0_y.

I can project f_x and f_y along z and keep everything nice if (f^0_x) or f_{xy} and (f^0_y) or f_{xy}. Now I have f^+_{xx}, f^+_{yy} and I want to think about the locus where f^0_x. It's still single-valued with respect to x, so project f_{yy} to $f_x = 0$ along x, maintaining its y-derivative. This succeeds if $[f^0_{xy}]$ or $[f^0_{xy}]$. Then we find a unique point where f^0_y.

The four-dimensional cases:

(The Typical 4-d case)

Suppose I encounter $f^+_{xx}, f^+_{yyy}, f^+_z, f^+_{wy}$ and f^0_x, f^0_y, f^0_{yy}.

Project to f along z. Okay if $(f^0_{xz}$ or (f^0_x), f^0_{yyz} or (f^0_y), $[f^0_{yz}]$ or $f^0_w)$

Project to f_x along x. Okay if $(f^0_{yyx}$ or $[f^0_{xy}]$, $[f^0_{xy}]$ or $f^0_w)$

Project to f_y along w. Okay if $(f^0_{yyw}$ or $(f^0_{yy}))$

Project to f_{yy} along y. Okay, because no further derivatives need to be preserved.

(two other cases isomorphic to this one, I think)

(*Adjoint Case*)

Suppose I encounter $f^+_{xxxx}, f^+_y, f^+_{xz}, f^+_{xxw}$ and $f^0_x, f^0_{xx}, f^0_{xxx}$.
Project to f along y. Okay if (f^0_{xxxy} or (f^0_x), [f^0_{xy}] or f^0_z, [f^0_{xxy}] or f^0_w)
Project to f_x along z. Okay if (f^0_{xxxz} or (f^0_{xx}), [f^0_{xxz}] or f^0_{xw})
Project to f_{xx} along w. Okay if (f^0_{xxxxw} or (f^0_{xxx}))
Project to f_{xxx} along x. Okay, because no further derivatives need to be preserved.

(*Equivalence Case*)
Suppose I encounter $f^+_{xx}, f^+_{yy}, f^+_{zz}, f^+_w$ and f^0_x, f^0_y, f^0_z.
Project to f along w. Okay if (f^0_{xw} or (f^0_x), f^0_{yw} or (f^0_y), f^0_{zw} or (f^0_z))
Project to f_x along x. Okay if ([f^0_{yx} or f^0_{xy}], [f^0_{zx} or f^0_{xz}]???)
Project to f_y along y. Okay if ([f^0_{yz} or f^0_{zy}])
Project to f_z along z. Okay, because no further derivatives need to be preserved.

Crap, this last one required f^0_{xz}, which is not forbidden throughout the entire tree, — but it is ruled out in any path that begins with a right branch.

Let me try the last one:
(*The xz^2 case*)
Suppose I encounter $f^+_{xxx}, f^+_y, f^+_{xzz}, f^+_{xw}$ and $f^0_x, f^0_{xx}, f^0_{xz}$.
Project to f along y. Okay if (f^0_{xxy} or (f^0_x), [f^0_{xyz}] or f^0_z, [f^0_{xy}] or f^0_w))
Project to f_x along w. Okay if (f^0_{xxw} or (f^0_{xx}), f^0_{xzw} or (f^0_{xz}))
Project to f_{xx} along x. Okay if ([f^0_{xxz} or f^0_{xxz}])
Project to f_{xz} along z. Okay, because no further derivatives need to be preserved.

(**TW**) Thought about proof of adequacy of relevant HLF. It's somewhat tricky, since the translation is no longer strictly monotone in information.

(**TFU**) Duplicated hlf code to start working on relevant unification and coverage tests.

(**TFC**) Poked at kolm bug.

(**TE**) Still have gas bill, need to get receipts.

2008.11.30

I think it might be important that I can do projections in any order to pull out information about domains and codomains.

(**TW**) Thought more about proof of adequacy of relevant HLF. The thing to do is set up two versions of LLF and two of HLF, which differ in synthesis vs. checking of worlds.

(**TFU**) Compiled.

(**TFC**) Looked at trace more.

(**TE**) Still have gas bill, need to get receipts, need to ask steph when they're heading up to dad's.

2008.12.1

(**TW**) Edits.

(**TFU**) Hacking on translation and unification for 'relevant' HLF. The next things to do are unification cases for $*$, and translation of terms — be sure to get lambdas and caret.

(**TFC**) Hacking on HLF examples.

(**TE**) Paid gas bill.

2008.12.2

(**TW**) Edits.

(**TFU**) Hacking

(**TFC**) Hacking

(**TE**) Need to pay rent, get receipts, call steph.

Suppose I have some function $k(x, y)$ that satisfies $k(x, y) = 1 - k(y, x)$ and monotone in its first argument, meant to represent the probability that a game player of 'strength' x beats a player of strength y.

Given that I observe ρ_{ij} instances of player i beating player j, what is the assignment of strengths σ_i to players that maximizes the likelihood

$$\prod_i \prod_j k(\sigma_i, \sigma_j)^{\rho_{ij}}$$

or equivalently maximizes the log likelihood

$$\sum_i \sum_j \rho_{ij} \ln k(\sigma_i, \sigma_j)$$

Take the partials

$$\frac{\partial}{\partial\sigma_i}\sum_i\sum_{j<i}\rho_{ij}\ln k(\sigma_i,\sigma_j)$$

$$=\frac{\partial}{\partial\sigma_i}\sum_{j\neq i}\rho_{ij}\ln k(\sigma_i,\sigma_j)+\rho_{ji}\ln k(\sigma_j,\sigma_i)$$

$$=\frac{\partial}{\partial\sigma_i}\sum_{j\neq i}\rho_{ij}\ln k(\sigma_i,\sigma_j)+\rho_{ji}\ln(1-k(\sigma_i,\sigma_j))$$

$$=\sum_{j\neq i}\rho_{ij}\frac{k'}{k}+\rho_{ji}\frac{-k'}{1-k}$$

$$0=\sum_{j\neq i}k'\left(\frac{\rho_{ij}}{k}-\frac{\rho_{ji}}{(1-k)}\right)$$

Suppose $k(x,y)=x/(x+y)$ then $k'=k_x=y/(x+y)^2$.

$$0=\sum_{j\neq i}\frac{\sigma_j}{(\sigma_i+\sigma_j)^2}\left(\frac{\rho_{ij}(\sigma_i+\sigma_j)}{\sigma_i}-\frac{\rho_{ji}(\sigma_i+\sigma_j)}{\sigma_j}\right)$$

$$0=\sum_{j\neq i}\frac{\sigma_j}{(\sigma_i+\sigma_j)}\left(\frac{\rho_{ij}}{\sigma_i}-\frac{\rho_{ji}}{\sigma_j}\right)$$

$$0=\sum_{j\neq i}\frac{1}{(\sigma_i+\sigma_j)}\left(\rho_{ij}\frac{\sigma_j}{\sigma_i}-\rho_{ji}\right)$$

If I set $\tau_i=\sigma_i^{-1}$ then

$$0=\sum_{j\neq i}\left(1+\frac{\tau_i}{\tau_j}\right)\left(\frac{\rho_{ij}}{\tau_j}-\frac{\rho_{ji}}{\tau_i}\right)$$

$$\sum_{j\neq i}\frac{\rho_{ij}}{\tau_j}-\frac{\rho_{ji}}{\tau_i}=-\sum_{j\neq i}\left(\frac{\tau_i\rho_{ij}}{\tau_j^2}-\frac{\rho_{ji}}{\tau_j}\right)$$

Consider how the statistics get distorted for 'best n out of m'. The better player is more likely to win. Does this provide a natural time scale for competition?

2008.12.3

Had a bug where I was was throwing away the substitution returned by whnf, assuming that it was always id. It's not! It's only in times where you get like a root back where the substitution must be id. Since evars don't wrap their own substitution, you might get back the eclo that is the evar and the subst.

Here's the counterexample regression test:

```
%hlf.

o : type. %name o A.
ca : {Q:w}
      ({a:w} o @ (a * Q))
      -> type.

ca_axiom_l : ca  _  ([x:w] E x).
```

The crazy symptom was a doublecheck failure where a reconstructed implicit argument of type w was nonetheless getting applied to two arguments.

Another mistake I keep making is pattern-matching on Root(BVar n, Nil). That Nil might still be an SClo!

(**TW**) Edits.

(**TFU**) Hacking

(**TFC**) Hacking

(**TE**) Need to pay rent, get receipts, call steph, respond to m4m5 email.

2008.12.4

Here is the judgment ordering, I think, if I want to include asynchronous decomposition:

Positive focus $\geq$ Positive inversion $\geq$ Negative inversion $\geq$ Negative focus

2008.12.5

Kind of gave up on representing ordered decomposition of asynchronous things, simply cleaned up the representation of synchronous.

Noam pointed me to a 1995 lics paper by Francois Lamarche that does something stunningly similar with token passing, but in the context of

double-negation translation. Surely I can say something about just plugging in the double-negation adjunction for F, U? What deeply perplexes me is his notion of polarity distinguishes conjunctions from disjunctions, and yet he claims it's the same as Girard's notion.

(**TW**) Edits.

(**TFU**) Unification: looked at Anders's thing. Looks great! It's exactly the pattern fragment I'd predict from HLF.

(**TFC**) Coverage: falling off the edge of the world in abstraction. Is this just because I don't actually do unification yet?

(**TE**) Responded to m4m5 email. Need to pay rent, get receipts, call steph, update papers webpage.

2008.12.6

Video effect idea: a bunch of crawling lines each of which looks independently like it is moving parallel to itself, but when they overlap and weave together they form an arbitrary video.

Efficient implementation is easy if you keep a buffer of source pixel (one per moving line) and phase offset, and prioritize the source pixel being 'here' if ever someone tries to draw over it.

(**TW**) Edits.

(**TFU**) Unification: ok, so I'm doing unification properly I think, and at least I'm postponing nothing, but still getting the abstraction error. Dunno what's up with that.

(**TFC**) Coverage: Thought more about positive connectives.

(**TE**) Paid rent. Need to respond to tcs email, get receipts, call steph, update papers webpage.

2008.12.7

Instead of making F and U so concretely about linear logic, I can think of them as being about an off-to-the-side number (also resembling a creation-annihilation pair)

$$\frac{\Gamma \vdash_n A}{\Gamma \vdash_{n+1} FA} \qquad \frac{\Gamma, A \vdash_{n+1} C}{\Gamma, FA \vdash_n C} \qquad \frac{}{A \vdash_0 A}$$

$$\frac{\Gamma \vdash_{n+1} A}{\Gamma \vdash_n UA} \qquad \frac{\Gamma, A \vdash_n C}{\Gamma, UA \vdash_{n+1} C} \qquad \frac{\Gamma \vdash_n A \qquad \Delta(A) \vdash_m C}{\Delta(\Gamma) \vdash_{n+m} C}$$

This means that focalizing, say, ordered and bunched logic seems pretty easy. One semi-surprising conclusion is that the bunched additive conjunction is ambipolar! But then I thought about it and I realized it is so for roughly the same reason that the unrestricted implication is: on the first day of creation we invented a context-constructor (comma in the case of unrestricted "plain" logic, semicolon in bunched) and we gave it an internalization (some kind of $\otimes$, really) and a right adjoint (some kind of $\multimap$). Also separately we invented negative conjunction. But on the second day we added structural rules to cause the positive and negative conjunction to be equiprovable!

It's less natural to see that this was the story in bunched logic, where if we strip away the structural properties, the positive conjunction that internalizes semicolon becomes just a funny different sort of comma. But I believe it's the right story to tell.

Ambifortunately I see that HLF seems to have the same property, that its additive conjunction is 'plain' conjunction, both positive and negative. So that even if I take the Church encoding of disjunction

$$A \oplus B = \forall \tau{:}*.(A \multimap \tau) \mathbin{\&} (B \multimap \tau) \multimap \tau$$

I'm bound to still get

$$A \mathbin{\&} (B \oplus C)@p \dashv\vdash A \mathbin{\&} B \oplus A \mathbin{\&} C@p$$

No, wait, what? To build the appropriate term left to right I need to consume the resource p to even get the branching required. Maybe this does work. What's going on, I don't know.

Okay, on the left I even know what type I'm eliminating at, it's $X = A \mathbin{\&} B \oplus A \mathbin{\&} C$, and for positive occurrences of τ I can take it to be a free type variable, so really

$$X = (A \mathbin{\&} B \multimap \tau) \mathbin{\&} (A \mathbin{\&} C \multimap \tau) \multimap \tau$$

So I've got

$$A \mathbin{\&} ((B \multimap X) \mathbin{\&} (C \multimap X) \multimap X)@p \vdash X@p$$

which looks like once I consume p I've only got ϵ left to prove A.

And yet I can't figure out how to refute the equivalence between the Church disjunction and the one that goes

$$\frac{\Gamma, A[p] \vdash C \qquad \Gamma, B[p] \vdash C}{\Gamma, A \oplus B[p] \vdash C}$$

Maybe one of them doesn't satisfy cut?

Okay, take the proof

$$\dfrac{\dfrac{\dfrac{}{A[p], B[p] \vdash A \,\&\, B[p]}}{A[p], B[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}}{\dfrac{A[p], (B \oplus C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}{A \,\&\, (B \oplus C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}} + sym$$

and cut against

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash (B \multimap B \oplus C) \,\&\, (C \multimap B \oplus C)[\epsilon]}}{(B \multimap B \oplus C) \,\&\, (C \multimap B \oplus C) \multimap B \oplus C[p] \vdash (B \oplus C)[p]}}{\forall\tau.(B \multimap \tau) \,\&\, (C \multimap \tau) \multimap \tau[p] \vdash (B \oplus C)[p]}\, id}{\dfrac{A[p], \forall\tau.(B \multimap \tau) \,\&\, (C \multimap \tau) \multimap \tau[p] \vdash A \,\&\, (B \oplus C)[p]}{A \,\&\, \forall\tau.(B \multimap \tau) \,\&\, (C \multimap \tau) \multimap \tau[p] \vdash A \,\&\, (B \oplus C)[p]}}$$

and I get

$$\dfrac{\dfrac{}{A[p], (B \multimap B \oplus C) \multimap (C \multimap B \oplus C) \multimap (B \oplus C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}}{\dfrac{A[p], (B \vee C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}{A \,\&\, (B \vee C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}}$$

and this seems to succeed.

Well, what's the natural deduction rule I expect?

$$\dfrac{\Gamma \vdash A \vee B[p] \qquad \Gamma, A[\alpha] \vdash C[\alpha * q] \qquad \Gamma, B[\alpha] \vdash C[\alpha * q]}{\Gamma \vdash C[p * q]}$$

I should fail to prove the sequent rule

$$\dfrac{\Gamma, A[p] \vdash C[q] \qquad \Gamma, B[p] \vdash C[q]}{\Gamma, A \vee B[p] \vdash C[q]}$$

sound then, right? Though I haven't reasoned through it yet, I think yes! The natural deduction rule it corresponds to is

$$\dfrac{\Gamma \vdash A \vee B[p] \qquad \Gamma, A[p] \vdash C[q] \qquad \Gamma, B[p] \vdash C[q]}{\Gamma \vdash C[q]}$$

Augh, but they share a right rule. That can't be right. How can I internalize the correct left rule?

$$A \vee B[p] = \forall C : *.\forall q : w.(A \multimap C)@q \to (B \multimap C)@q \to C@(p * q)$$

but this is equivalent to

$$A \vee B = \forall C : *.(A \multimap C) \,\&\, (B \multimap C) \multimap C$$

what is going on?

At least the elim rule without the epsilons makes the one that has them admissible by substitution. And I can see soundness of the standard sequent left rule with respect to the non-alpha elim rule.

So the α-ish left rule has to be like

$$\frac{\Gamma, A[\alpha] \vdash C[\alpha * q] \qquad \Gamma, B[\alpha] \vdash C[\alpha * q]}{\Gamma, A \vee B[p] \vdash C[p * q]}$$

and this, while it satisfies identity and rules out

$$\frac{\dfrac{}{A[p], B[\alpha] \vdash A \,\&\, B \vee A \,\&\, C[\alpha]}}{A[p], (B \vee C)[p] \vdash A \,\&\, B \vee A \,\&\, C[p]}$$

I'm worried it doesn't satisfy cut because of that $p * q$ in the conclusion. So let me again take the derivation

$$\frac{\dfrac{\dfrac{}{A[p], B[p] \vdash A \,\&\, B[p]}}{A[p], B[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]} \qquad + \, sym}{A[p], (B \oplus C)[p] \vdash A \,\&\, B \oplus A \,\&\, C[p]}$$

and try to cut in

$$\frac{\dfrac{}{B[\alpha] \vdash (B \oplus C)[\alpha]} \qquad + \, sym}{(B \vee C)[p] \vdash (B \oplus C)[p]}$$

Yes, there's a failure of a commutative case lurking there.

If I have

$$\frac{\dfrac{\Gamma, A_i[\alpha] \vdash D[\alpha * q]}{\Gamma, A_1 \vee A_2[p] \vdash D[p * q]} \quad \dfrac{}{\Gamma, D[p * q] \vdash C[r]}}{\Gamma, A_1 \vee A_2[p] \vdash C[r]} \, cut$$

I lose parametricity if I try to commute.

I want to figure out why the Church encoding doesn't effectively create this elim rule. Is the focusing discipline satisfied suitably? I seem to have no choices about how to polarize

$$A \oplus B = \forall \tau{:}*.(A \multimap \tau) \,\&\, (B \multimap \tau) \multimap \tau$$

and if I put it $@p$, then I suppose I am considering $@$ to be negative.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A_i[\alpha] \vdash D[\alpha * q]}}{\vdash A_i \multimap D[q]}}{\dfrac{}{D[p*q] \vdash D[p*q]} \quad \vdash (A_1 \multimap D) \,\&\, (A_2 \multimap D)[q]}}{(A_1 \multimap D) \,\&\, (A_2 \multimap D) \multimap D[p] \vdash D[p*q]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau[p] \vdash D[p*q]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau@p \vdash D[p*q]} \quad \dfrac{}{D[p*q] \vdash C[r]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau@p \vdash C[r]}\, cut$$

It looks as if I was cutting against the wrong thing — should be the $D[p*q] \vdash D[p*q]$, not the blob involving αs. But that $D[p*q]$ only arose as a lucky synchronous guess. What's the more general case?

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A_i[\alpha] \vdash D[\alpha * q]}}{\vdash A_i \multimap D[q]}}{\dfrac{}{D[p*q] \vdash E[s]} \quad \vdash (A_1 \multimap D) \,\&\, (A_2 \multimap D)[q]}}{(A_1 \multimap D) \,\&\, (A_2 \multimap D) \multimap D[p] \vdash E[s]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau[p] \vdash E[s]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau@p \vdash E[s]} \quad \dfrac{}{E[s] \vdash C[r]}}{\forall \tau : *.(A_1 \multimap \tau) \,\&\, (A_2 \multimap \tau) \multimap \tau@p \vdash C[r]}\, cut$$

Suggesting the sequent rule

$$\dfrac{\Gamma, A_i[\alpha] \vdash D[\alpha * q] \qquad \Gamma, D[p*q] \vdash E[s]}{\Gamma, A_1 \vee A_2[p] \vdash E[s]}$$

if D is a negative atom, then this forces E (since D it is still in focus) to be the same atom. But perhaps it isn't! Indeed, this rule allows me to 'cut'

through $A \oplus B$, in a way that is never essentially eliminated, allowing a proof of $A \,\&\, (B \vee C) \vdash A \,\&\, B \vee A \,\&\, C$.

So what I *really* want is a $\forall$ over all negative atoms. I'm still worried that it looks like it has the same right rule, but it's worth investigating. Especially because if I think of atoms as indexed zero-ary connectives, then the $\forall$ is ultra-predicative, i.e only quantifies over the index domain, not propositions at all!

On second thought, maybe the intro rules are different, and what's more correct. If asynchronously a disjunction tosses a hypothetical negative atom τ into the context and gives me $(A \multimap \tau) \,\&\, (B \multimap \tau)$ then every time I use those I have to use up that τ against the conclusion rather than keep it around. Strangely, $A \oplus B$ still exists, and I cannot prove $A \vee B \vdash A \oplus B$ (but I can prove the converse). To prove the identity $A \vee B \vdash A \vee B$ I must decompose the right to make it a negative atom, then decompose the left.

If I want to truly 'incorporate' $A \vee B$ I suppose it might go in some syntactic category that generalizes negative atoms.

This whole story should play out similarly with $A \wedge B$ and $A \otimes B$, I suppose.

$$\frac{\Gamma, A_i[\alpha] \vdash a^-[\alpha * q]}{\Gamma, A_1 \vee A_2[p] \vdash a^-[p * q]} \qquad \frac{\Gamma, A_1[\alpha], A_2[\beta] \vdash a^-[\alpha * \beta * q]}{\Gamma, A_1 \wedge A_2[p] \vdash a^-[p * q]}$$

Perhaps this is finally a justification for those funny 'frame elimination rules' I had so long ago.

If all positive connectives are *actually* second-order negative connectives in Church-disguise quantifying over negative atoms, then there's no obstacle to applying positive left rules at any time, since we can always by inversion get the right hand side to be an atom.

(**TW**) Thought about positives, made more progress.

(**TFU**) Unification: Hacking.

(**TFC**) Coverage: Hacking.

(**TE**) Talked to Jess about scheduling. Need to pack, cash checks, respond to black friday, tcs emails, get receipts.

2008.12.8

I should see what results I can extract from the self-adjunction

$$- \multimap p^- \dashv - \multimap p^-$$

noting that both functors are contravariant, so here I demand $A \to UFA$ and also $B \to FUB$.

Weirdly it seems one of them should still be positive — I suppose I have to squint at the convtravariance to make that go away.

Okay, so proof search inside that particular monad requires that I decompose a $\multimap p^-$ on the right, otherwise focusing on it on the left will fail. And if I do choose $N \multimap p^-$ on the left, I need to keep decomposing it until I get to the p^-, don't I? Hm.

Current type reconstruction bug is this:

```
%hlf.


sow : type.
reap : type.
seed : type.

sow/z : sow o- reap.
sow/s : sow o- (seed -o sow).
reap/z : reap.
reap/s : reap o- seed o- reap.

lemma1 : {A:w} sow @ A -> reap @ A -> type.
lemma : {A:w} {B:w} sow @ A -> reap @ B -> type.

lemma/s :
      ({a:w} {x:seed @ a} lemma1 _ (SOW _ x)
                           (reap/s _ REAP a x))
         -> lemma1 _ (sow/s _ SOW) REAP .
```

It generates epsilon for the world the lemma is working at, rather than a free world variable. If I set (SOW _ x) instead to (SOW a x), then it's fine.

I think I'm going to plow ahead and see if I can get coverage for a more explicitly-typed version.

Made some solid progress!

Bugs apart, I should probably prioritize uniqueOccur over singleton checking in unification; uniqueOccur can dispatch some cases that result in postponed constraints otherwise.

I notice coverage, when splitting on worlds, picks up all the garbage artificial "world constructors" at the beginning of the signature. Should fix that.

The main thing is I don't anticipate actually using the world index on metatheorems and it's just causing problems. All the worlds should be quantified *arguments* to the metatheorem, where they have sensible input/output status.

(**TW**) Thought more about positives.

(**TFU**) Unification: Hacking.

(**TFC**) Coverage: Hacking.

(**TE**) Cashed checks. Need to pack, respond to black friday, tcs emails, get receipts.

2008.12.13

The Nullstellensatz says that the comonad arising from the adjunction $I \dashv V$ (where I computes the polynomial ideal of a subset of $\mathbb{A}^n$, and V computes the variety of an ideal) is exactly $\sqrt{\text{—}}$.

$$p \in IS \Leftrightarrow S \subseteq V(p)$$

2008.12.15

I've returned for some reason to thinking about string diagrams in terms of piecewise linear approximations.

Say a *signature* Σ is a collection of sets $(S_i)_{i \in \mathbb{N}}$. Then an *n-diagram over* Σ is a map d from $\mathbb{R}^n$ to $\coprod_i S_i$ such that for every point $x \in \mathbb{R}^n$ that d takes to a label $\ell \in S_i$, there is a $U \ni x$ for which $f^{-1}(\ell) \cap U \cong \mathbb{R}^i$ — specifically we mean the inverse image of that label in that neighborhood is a linear subspace of the local linear space centered at x of dimension i — and for every $\ell' \in \{S_j \mid j \geq i\}$ other than ℓ, we have $f^{-1}(\ell') \cap U = \emptyset$.

Then a notion of category should be some restriction on these labellings that only permit diagrams that are "well-domained".

Say the class of diagrams is called D. A category is a mapping k from $\coprod S$ to D for which the 'local picture' at the origin is always equal to the 'global picture', and which at the origin yields the label that was fed into k in the first place.

A category as we know it is one that is equipped with enough 2-cells that 1-cells can go 'in any direction they like' as long as they don't reverse in 2-time.

We depart from the free category on a graph to the extent that extra 2-cells exist, postulating facts about composition.

I wonder how one could be agnostic about the *primitivity* of cells — if I stick in a 2-cell that allows composition of two 1-cells to yield a third, then it necessarily does so 'at a particular angle'. And when I'm in a setting that allows arbitrary bending of 1-cells, then 'molecular' 2-cells always exist that allow me to compose at all other angles — but these aren't 'primitive'.

2008.12.17

Here's another angle.

Given an n-category, the collection of string diagrams one can draw using its cells in a varying open subset of $\mathbb{R}^n$ is a sheaf.

But it actually has more structure! We can restrict our attention to diagrams defined over the box $B_n = \prod_n (-1, 1)$. The notion of restriction in the presheaf over open sets is replaced by 'restriction-and-zooming-in', an operation that takes a diagram over B_n and some other box $b \in B = \{\prod_{0 \leq i < n} (a_i, b_i) \,|\, -1 \leq a_i < b_i \leq 1\}$ to a diagram over B_n.

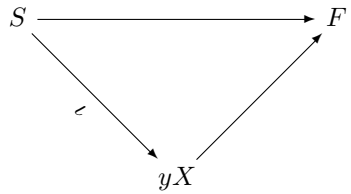
We can then require that structure to satisfy functoriality and pasting axioms like we would a sheaf.

Actually, I think what I want is sheaves for a certain Grothendieck topology. The category has one object, and morphisms are functions that take B_n to itself in some nice rectilinear shrinking way. A covering sieve is just one where the union of the ranges of all the maps in it covers B_n.

Wikipedia gives the condition for a presheaf $F : \mathbf{Sets} \to \mathbf{C}^{op}$ to be a sheaf on a site $(\mathbf{C}, J)$ to be the following: Let an arbitrary object $X \in \mathbf{C}$ and a covering sieve S on X from the topology J be given. Now S, being a sieve, is a subpresheaf of what we get if we hit X with the Yoneda embedding, yX. So we have an arrow $\iota : S \to yX$. Well, the hom functor for the category of presheaves is contraviarant in its first argument, so we have a function $Hom(\iota, F) : Hom(yX, F) \to Hom(S, F)$. For F to be a sheaf, this is supposed to always be an isomorphism.

Oh! Suddenly I remember how this usually looked diagrammatically. It

was unique factorization of any arrow $S \to F$ through ι:



Now I should like from any sheaf on this site to construct a category, and in the future more generally an n-category.

I think this proceeds by saying that the highest-dimensional cells are straightforwardly elements of the sheaf at its only object, and the lower-dimensional cells are arrived at by 'germ'-like constructions.

Like say for any cell x in the sheaf, the 'symbols' $\operatorname{dom} x$ and $\operatorname{cod} x$ are candidates — maybe in fact I should call them *cocandidates*, since they're not going to be filtered but rather quotiented — for being 1-cells.

We say that any restriction of a diagram x 'towards its domain' is a *future* for $\operatorname{dom} x$ and likewise any restriction towards the codomain of x is a future of $\operatorname{cod} x$.

Equate any two cocandidates that share a future.

Here's where I might start imposing conditions on sheaves to get them to be exactly the sort of things that arise from considering the string diagrams on a category: I would want that the domain and codomain of a cell themselves have a coherent domain and codomain, and that the usual conditions $cd = cc$ and $dd = dc$ hold.

I notice that in the encoding of a category, 0-cells are thoroughly 'self-similar' in that restrictions anywhere leave the diagram fixed. 1-cells have a weaker property that no matter how much 2-time you knock out, there is some point remaining that looks like the whole. Moreover there is a certain about of 1-time you can strip away at either side so that what's left is a 0-cell.

If I think about a 2-cell embedded in 3-time, then again I can knock out as much 3-time as I like and there is still the same 2-cell remaining, somewhere.

2008.12.19

Some further observations:

Say the one object of the site is called X. Take a sheaf C. Say that

$x \subseteq y$ ('x embeds in y') for $x, y \in C(X)$ if there exists an $f : X \to X$ such that $y\big|_f = x$.

For the sheaf of diagrams of a category, mutual embeddability of two diagrams seems like a very strong condition, even without demanding that the composite of the two embeddings is somehow 'the identity' (and I wouldn't even know what condition to impose to represent that — the site's morphisms seem to 'run only in the one direction' of creating finer and finer embeddings, and don't approach the identity in any obvious sense)

So take the equivalence relation $x \sim y$ to be $x \subseteq y \wedge y \subseteq x$.

(incidentally: did I already mention the fact that it's easy using the generality of Grothendieck topologies to account for *finite* diagrams? For all I need to do is restrict the sieves in the topology to those that can be built as finite unions of principal sieves)

Now certain morphisms $f : X \to X$ need to be selected out as representing n-temporal domain and codomain. They are going to be the things that look like restricting to and zooming in to the n-time beginning or end of a cell. Suppose I have sets $C_n, D_n \subseteq Hom(X, X)$ given to me. Let $B_n = \bigcup_{m \geq n} C_m \cup D_m$.

Having done that, for any $S \subseteq Hom(X, X)$ say a diagram x is *stable for* S if $x\big|_f \sim x$ for any $f \in S$.

Here is a property I probably want to demand: informally, that every cell has an n-domain and n-codomain. Formally, for every $x \in C(X)$ and $n \in \mathbb{N}$, there exists $f \in C_n$ such that $x\big|_f$ is stable for B_n, and likewise for D_n.

2008.12.21

Playing around with emacs stuff. Thinking about allowing editing of BDFs by just writing some hex-ASCII conversion and using picture mode since getting Perl/Tk to work on cygwin is surprisingly painful.

I wonder, is the common trope of emacs motions functions allowing integers or things-that-represent-integers like marks somehow an example of comonadic programming?

2008.12.22

Ingrid Michaelson sounds kinda like Regina Spektor, and Hello Saferide like Ani. Something in the r's.

The emacs bdf editing works well enough to fix my broken first draft of codon italic.

2008.12.23

PERs feel special because they are a coequalizer of an equalizer; a quotient of a filter. Or, symmetrically, the other of the one in both cases, I think. How does this generalize? What process got me from the category

of Sets to the category of PERs?

Arrows that are the equalizer (resp. coequalizer) of a parallel pair are by definition regular (epi) monomorphisms (resp. epimorphism). So do I just want to ask for a diagram like the following?

$$A \xrightarrow{f} B \xrightarrow{g} C$$

Maybe so. That actually looks fairly correct. Anything not in the image of g are things that fall outside the PER, and A is divided up by f into stalks of equivalent elements over B.

What's missing from this picture is the original 'underlying set' over which realizability is defined. Like if we have a computational process telling us when particular Gödel numbers are equivalent, then it's doing that over $\mathbb{N}$. So maybe the full picture is

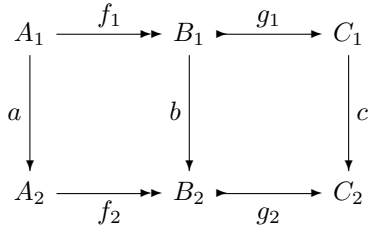
$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow & & \downarrow g \\ \mathbb{N} & \longrightarrow & C \end{array}$$

Or in fact the other side of the diagram is also mono-epi, but in the opposite order:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow & & \downarrow g \\ \mathbb{N} & \longrightarrow & C \end{array}$$

In Sets I can recover one side of this diagram (up to isomorphism) from the other: $\mathbb{N} \cong A \cup (C \setminus B)$. The converse seems sketchy, because not every lower-left path in such a diagram automatically looks like a PER; we could nontrivially quotient out some of those things that aren't even in the image of A. I guess I want the 'complement' of the diagonal from A to C, whatever that means, to be mono?

I'll retreat to the supposition that the original unbent diagram is what I want. Perhaps a morphism from one 'PER object' to another is what you'd expect:



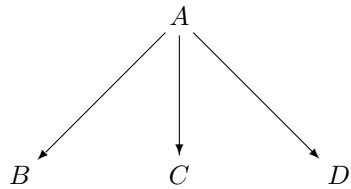
Are these exactly equivariant maps? Not quite. It seems to equate functions f, g who differ up to equivalence on irreflexive elements. That is, if $x \not\sim x$ and $f(x) \sim g(x)$, then this is no obstacle to $f = g$.

But I think if I talk about morphisms over the full square diagram then this blurriness disappears; morphisms can be distinguished as maps on the 'full space' $\mathbb{N}$. So what I really want to do is identify the condition on commutative squares that means that $\mathbb{N}$ is a coherent choice as underlying set.

2008.12.26

Read Baez et al's recent paper on (de)groupoidification of linear algebra.

I left a comment on the n-Category Café about wondering what happens with groupoidifying tensors more general than just vectors and matrices. A multilegged span

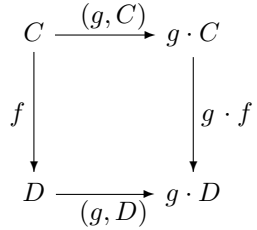


sure looks like just taking the three coordinates of some 3-d tensor in a chosen basis.

Also: can I make sense of a group g acting on a groupid $\mathbf{C}$ (yielding a groupoid) in such a way that $|\mathbf{C} /\!/ G| = |\mathbf{C}|/|G|$ and also recover the usual notion of group acting on a set yielding a groupoid by viewing the set as a discrete groupoid?

I suppose the action might take an element of G and a morphism of G and return, what, an object of G? That's not enough to state associativity of the action. Well, what is an action normally? It's a group homomorphism from g to the automorphism group of the set in question, of course. So I should demand a homomorphism $\cdot$ from G to $Aut(\mathbf{C})$.

What does the (weak) quotient look like? For backwards compatibility the objects are probably the objects of G, and the morphisms are $\{(g, C) \mid g \in G, C \in \mathbf{C}\}$ with $\mathrm{dom}(g, C) = C$ and $\mathrm{cod}(g, C) = g \cdot C$. Perhaps require naturality



But wait — did I also include the original morphisms from $\mathbf{C}$? Maybe so. I also need identities even after I quotient out a set by a group action.

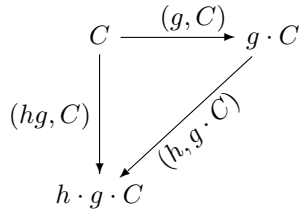
No, I already have the action of the identity element of the group to account for that.

Oh! And likewise, I still have $e \cdot f$ to represent the inclusion of the old groupoid in the new. Okay, good. This diagram is actually well-formed. It represents not a *requirement* of commutativities, but an *imposition* of them on the otherwise free addition of morphisms (g, C). Perhaps I need to add more still?

Arg, no, I still don't see the compulsion to include $g \cdot f$ for any g necessarily. Maybe it's just what needs to be done.

Got to settle on a equational theory on these paths if I'm to conjecture a Burnside property $|\mathbf{C} /\!/ G| = |\mathbf{C}|/|G|$ holds in any definite sense.

I probably want



2008.12.27

It's easy to see that groupoid cardinality is preserved if one realizes it's equivalently defined by

$$|\mathbf{C}| = \sum_{C \in \mathbf{C}} |\mathrm{hom}(C, —)|$$

Apparently this whole business is just taking weak colimits.

I like how `allout` exploits funny little niches in communicative space. One, that things like `C-a` are idempotent so there's no need to execute it twice, so bouncing it several times is put to other use.

Two, it's unlikely that you want edit the bullet-point lines using ordinary text insertion, so it uses those for motion.

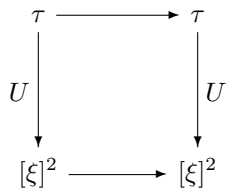
2008.12.27

When trying to simulate sexps in XML, one 'wastes resources' in describing data structures that are not basically marked-up text. When simulating XML in sexps, one 'wastes resources' in describing structures that are.

Emacs has both text properties and overlays: the former feel like the 'semantics' in the old ICFP programming contest problem about HTML optimization, and the latter feel like (a non-hierarchical version of) the markup language itself.

The image I have is something like this:

A plain, unformatted buffer-with-point is obviously the derivative of $[\alpha]$ (i.e. α list) evaluated at the type of characters, call it ξ. So it's $[\xi] * [\xi]$. Formatted text is a type sitting over this, with some kind of 'forgetting map' $U : \tau \to [\xi] * [\xi]$, which supports operations like insertion and deletion:



Where do insertion and deletion come from, conceptually? If I'm looking at the derivative of a monad, insertion makes sense given the monadic multiplication — it's the interpolative replacement of the hole with some term-with-hole. I suppose deletion is just the opportunistic inversion of such a process — kind of like how the annihilation operator can fail if not preceded by a creation.

I'm reminded of Benoît's point that if you create something and communicate it to no one, then you might as well not have created it — which I might modify by allowing that it might have some intrinsic value to you yourself, but admit that then it dies with you.

Read Cosma Shalizi's review of NKOS. It's a delicious bit of Schadenfreude. The thing he (I think rightly) accuses Wolfram of doing repeatedly is being technically right (that complex systems can arise from simple rules, that CA are kinda neat, etc.) but making the mistake of thinking an idea far more important than it really is, and failing to acknowledge others' significant working-out of that same idea. Shalizi writes:

> Wolfram refers incessantly to *his* "discovery" that simple rules can produce complex results. Now, the word "discovery" here is legitimate, but only in a special sense. When I took precalculus in high school, I came up with a method for solving systems of linear equations, independent of my textbook and my teacher: I discovered it. My teacher, more patient than I would be with adolescent arrogance, gently informed me that it was a standard technique, in any book on linear algebra, called "reduction to Jordan normal form", after the man who discovered it in the 1800s. Wolfram discovered simple rules producing complexity in just the same way that I discovered Jordan normal form.

I think I make this species of discovery all the time, and I'm not sure what to do about it. On the one hand, it would be nice for each such discovery to find other people's thinking on it, but on the other hand, it's hard to map from 'vague reinvention in private notation' to 'canonical definition used by broad mathematical tradition' without using a human being, which is costly and, let's be frank, potentially embarrassing.

One thing about the way ML handles datatypes (and from what I dimly remember, the way O'Caml handles record types) is that it establishes a unique mapping from the name of a constructor to the name of the type it constructs. This is directly antithetical to any attempt to have polymorphic variants in some sense, but it's obviously useful for type inference.

Could one achieve some sort of hybrid where one can specify different names for the *same* constructor, each of which 'suggests' a different refinement of the same type?

The creation and annihilation operators are just d/dx and x on *exponential* power series (i.e. those with extra $1/n!$ coefficients)

One possible use for 'two-use' linear variables in HLF is representing undirected graph edges.

2008.12.29

(**TW**) Reboot

(**TFU**) Reboot

(**TFC**) Reboot

(**TE**) Need to deal with tcs emails, get receipts.

2008.12.30

A confounding constellation of questions for tag systems: There are situations where there are three natural classes x, y, z where I might have good reasons for tagging them as $A, A \wedge B, B$ or A, C, B where C effectively means $A \wedge B$, or else I might do $D \wedge C, C, B$ where D means effectively $\neg B$.

Actually these examples are a bit sloppy.

Consider a space $\Omega = A \cup B \cup C$, where A, B, C are disjoint. Suppose all my data only needs to be labelled as $AB = A \cup B$ or $BC = B \cup C$ or B. Then in that case, if the semantics of multiple tags is conjunctive, I need only AB and BC as primitive tags.

But wait, interpreting multiple tags as conjunctive is not at all obvious. Is it? I can't at all represent A here, and I can't represent AB given A and B.

Retreating a moment to the intended semantics, the conjunctive interpretation comes from treating a tag of T as meaning 'this text has something to do with T' i.e. belongs to the set 'has something to do with T'. In that sense, if it has something to do with many topics, then I'm just taking the conjunction.

The ambiguity I'm hunting after has to do with the fact that the set of *searches* is very probably not limited in any reasonable system to exact tags. Searching on expressions involving conjunctions and negations should be easy.

Incidentally, why does search always seem to come back to classical logic? What could 'constructive search' mean? Is it just a restriction of ordinary search where I don't assume that an untagged item definitely lacks that tag? Perhaps so. Then it would make sense also to *label* items with complex expressions, as if they were propositions known to be true at that world.

In fact I notice there's an asymmetry of choice in the contsructive setting whether we think of assuming the query and proving the proposition 'at the world', or vice-versa. In the classical setting, assuming $A \wedge \bigwedge \Gamma$ and proving A lines up with assuming A and proving $A \vee \bigvee \Gamma$.

A more general thing is to suppose the text has associated with it a proposition P, and the query is some function F from props to props. Search includes an item if $F(P)$. Restricting search to substitutive functions seems to make plenty of sense. Then the above two scenarios are just $\lambda X.X \Rightarrow Q$ and $\lambda X.Q \Rightarrow X$ for query Q.

(**TW**) Edited introduction.

(**TFU**) Recall that I suspected the problem had to do with pruning.

(**TFC**) Recall that much debugging output was dubious, because of naming of LVars.

(**TE**) Sent email about receipts. Need to deal with tcs emails.

Songs I enjoyed on dulcimer on YouTube: 'Rocky Top', 'Hard Times'

2008.12.31

Further thoughts on 'constructive' as opposed to classical search. I conjecture that the expressive power of 'contravariant' and 'covariant' search differ. What I mean by this is as follows: let D be a database of propositions, and Q be a query. Contravariant search returns the subset $Q^*D = \{d \in D \mid d \vdash Q\}$, and covariant search returns $Q_*D = \{d \in D \mid Q \vdash d\}$.

I'd say contravariant search can be faithfully embedded in covariant if there is a transformation f that computes a proposition from another one (and lifts to databases in the obvious way) such that $f(Q)^*f(D) \Leftrightarrow Q_*D$, and similarly vice-versa. Just for the sake of making a bet, I'd bet that one or the other direction lacks a faithful embedding, but really, I could see it going either way. It would be quite nice if both exist, I'm just not that optimistic.

Actually, I needn't demand that f is used both on the database and the query. Could be a different function for each.

If I allow linear logic, then they're equivalent! Let $f(X) = X \multimap p$ for a fresh atom p. Assuming everything else is nonlinear, this calls into question decidability. But if my nonlinear stuff stays in the nonlinear fragment, then I needn't worry about clogging my linear context with an unknown amount of stuff — focusing should probably tell me that right away it gets consumed and heads back to the (idempotent!) nonlinear context. Dyckhoff techniques might then still work.

The same sort of trick might be doable with Pfenning-Davies style modality, or maybe multimodality might be required to get a closed system.

The question I'm really asking is about functions that take two propositions and return one. That is, which ones allow simulations of which others? This is a natural generalization of the question of expressivity of the single-place relation of provability in various logics, and has obvious generalizations in turn.

A nice idiom similar to $X \multimap p$ is making up an atom o (for 'obsolete') to tag an item as X but only 'weakly' or 'obsoletely' or 'deprecatedly' so: $o \to X$. In a database full of $o \to X$, searching (contravariantly) for X won't find it, but searching for $o \to X$ will.

The tag $X \vee Y$ is similarly interesting; it will fail to show up on a (contravariant) search for X or Y, but it will for a search on $X \vee Y$. It's as if (reminding me of the topological semantics of intuitionistic logic) it lives in the closure of X's search results and Y's search results.

One could build a small library L of premises that are included in every search implicitly, so that Q becomes $L \Rightarrow Q$. The library might include inclusion or equivalence facts between tags that were decided upon after tags were already deployed.

Heck, I could even imagine a reddit- or del.icio.us-like site that showed you the proof-term it generated for each result so that you could agree or disagree with particular rules! Such a thing is probably unimaginable overkill for most people's (probably even my) needs, but it's a lovely daydream to have.

2009.1.1

A simple observation about tree structures.

An address in an n-ary tree is given by a finite sequence of numbers $a = (a_0, \ldots, a_n) \in \mathbb{N}$, and say $|a|$ is the length of a.

I can transform one address to another via the operation $a * n$, defined by

$$|a * n| = n + 1$$

$$(a * n)_i = [i < |a|](a_i + [i = n])$$

(where the indicator $[P] = 1$ if P is true, and 0 otherwise)

Claim: any set of addresses that actually forms a tree can be obtained by repeated use of this operation starting with the empty address. If we don't mind empty nodes, then any set of addresses.

Let's think of this rather as a sequence of instructions $n_1, \ldots, n_k$, which generates $(\epsilon), (\epsilon * n_1), (\epsilon * n_1 * n_2), \ldots, (\epsilon * n_1 * \cdots * n_k)$. We could also include instructions I_m for $m \in \mathbb{N}$ so that the pair of instructions I_m, n is

by definition equivalent to $m + n, I_m$, and D_m providing that $D_m, m + n$ equivalent to n, D_m.

2009.1.2

Frank mentioned the LICS deadline is coming up. Page limit is 10 pages, two-column. I think I'm going to try for it.

2009.1.3

Looking at my old hylo paper, it (a) is 15 pages, single-column and (b) makes some unfortunately strong claims about the lack of applications for the relevant version of the theory. Oh well.

2009.1.4

Thought a little about Frank's comment that worlds might be useful for expressing things about primitive recursion over hoigher-order data. I think what I really should do is look at Brigitte's stuff more closely and then see if I can merely substitute out polymorphism-over-contexts with polymorphism-over-worlds.

Apart from that, I smell the usefulness of keeping linearity as a concept in the system (and only subsequently 'backing off' to allowing contraction and weakening in a controlled way) so that one could plausibly do case analysis on *whether* a term really uses a world-resource or not. For otherwise I'm not sure where the case analysis that susses out the variable case actually takes place.

2009.1.5

15 pages single column becomes 8 pages two-column, holy crap. I have plenty of room to work in, I guess.

So it looks like I did cover dependency just fine. What's left to talk about? I suppose I am describing the relevant version here. It should simplify the description of coverage and so on. I should be able to talk a little bit about the pattern fragment, too, I suppose, mention how it winds up being equivalent to Anders's definition.

Why am I doing the relevant version? It makes the metatheory simpler.

Is it just a matter of adding proof irrelevance later? No, it doesn't seem to be. For I am quotienting out across multiple choices of worlds, subject to the constraint that they 'add up' correctly, introducing a nonlocal interaction that is very difficult to cope with.

I am troubled by the choice of whether to do it in spine form or not. On the one hand, that is where all my proofs really live. I don't know that they work that well otherwise. On the other hand, it's a novelty that gets in the way of the fundamental ideas.

Likewise the choice of how to tease of the relevance of the worlds, I

guess? Between attaching it to $\top$ and attaching it to application.

Applications

- Substructural Nominal Logic
- Reasoning about Linear Sequent Calculus
- Substructural Operational Semantics
- Embed Linear Logic, LLF, Bunched Logic
- Pattern fragment?

Angles of Attack

- Hybrid Logic is cool, let's add it to something
- Logical Frameworks are cool, let's add something to them
- Linear Logic is cool, how can we reason about it?
- Kripke semantics are cool, can we do them for linear logic?

Embeddings

- LLF without $\top$
- 'LLF_r' comes in two flavors, one bidirectional (worlds at application) one synthesizing (worlds at top)
- Bunched logic doesn't get everything
- Maybe can get positives with atomic Church encoding. I feel like the negative polarity of the atom can't change provability, but it's probably essential for the proof going smoothly.

Priorities

- One can think about linear logic through the lens of hybrid logic
  - What does this get me? A solution to the problems of making a meta LLF
  - What's hybrid logic?
  - What's linear logic?

Outlining like this seems to be spinning my wheels. I just keep thinking of different ways of organizing everything, and make no progress until I sit down and start writing sentences.

2009.1.6

(**TW**) Edits.

(**TFU**) Poked at code

(**TFC**) Poked at code.

(**TE**) did TCS review, paid rent and water and cell phone, scheduled advisor meeting, need to: pay electric, gas, phone, give receipts to jen

2009.1.7

I think the right lemma for showing permutation preserves length is like

$$size\ \alpha\ M^+ \to size\ (\alpha * \beta)\ P^+ \to size\ \beta\ N^- \to plus\ M\ N\ P^- \to \text{type}$$

or maybe with the arguments to plus reversed.

2009.1.8

I still want to think about unification as some sort of funny judgment where the 'propositions' are term constructors and they come in pairs in an appropriate sense.

So that $f(A, B) \doteq f(C, D)$ becoming $A \doteq C, B \doteq D$ has something to do with $\otimes$. The fact of $(A, B) \doteq (C, D)$ making that transition does all the more directly. The other step going on is just $fA \doteq fB$ becoming $A \doteq B$, and ciriticially $fA \doteq gB$ becoming bottom. This synchronization is the mysterious thing about unification. Were it not for it I could just go on saying a bunch of boring homomorphismish things like probably even $A \Rightarrow B \doteq C \Rightarrow D$ maps to $(A \doteq C) \Rightarrow (B \doteq D)$.

Though I feel like I'd want some type discipline to constrain the propositional bits to match up unconditionally: maybe there would be a coercion [—] of equations into propositions, and then a pair of identical-up-to-[bracketed]-components propositions could be turned into a proper equation.

Actually, no, the proof irrelevance occurs at constant application, not at equations. Something like:

$$\frac{A \text{ prop } A' \qquad B \text{ prop } B'}{A \star B \text{ prop } A' \star B'} \qquad \frac{A \text{ prop } C \qquad B \text{ prop } C}{[A \doteq B] \text{ prop } [C \doteq C]} \qquad \frac{A \text{ prop } A'}{f(A) \text{ prop } \bullet}$$

with rules like

$$\frac{\Gamma, A \doteq B \vdash J}{\Gamma, [A \doteq B] \vdash J}$$

And I can either directly have rules like

$$\frac{\Gamma, A_1 \doteq B_1, A_2 \doteq B_2 \vdash J}{\Gamma, A_1 \otimes A_2 \doteq B_1 \otimes B_2 \vdash J}$$

or else consider $(A_1 \otimes A_2) \doteq (B_1 \otimes B_2)$ to be an abbreviation for $(A_1 \doteq B_1) \otimes (A_2 \doteq B_2)$ — that is, unification is (up to a point) a function on a pair of propositions of the same shape.

Aw, crap. Trying to prove that little theorem about sow/reap I found a bug in HLF.

```
%hlf.

nat : type.
s : nat -> nat.
z : nat.

plus : nat -> nat -> nat -> type.
plus/z : plus z N N.
plus/s : plus (s M) N (s P)
  <- plus M N P.

size : nat -> @type.
size/e : size z.
size/* : size (s z) -o size N -o size (s N).

subtract : size M @ A -> size P @ (A * B)
    -> size N @ B -> plus M N P -> type.
subtact/e : subtract size/e SIZE SIZE plus/z.
subtract/* : subtract (size/* ^ H ^ TL) SIZE1 SIZEOUT PLUSOUT
   <- subtract TL SIZE1 SIZE2 PLUS1.
```

It fails reconstructing the last clause on doublecheck.
I can shrink it to

```
size : @type.
size/e : size .
size/* :  size -o size  -o size .
```

```
subtract : size @ A -> size @ (A * B) ->  type.

subtract/* : subtract (size/* ^ H ^ TL) SIZE1
   <- subtract TL SIZE1.
```

I thought it had to do with an XXX in unification where I don't check for occurrences underneath a star, but apparently it's worse than that.

With just

```
%hlf.
o : type.
subtract : {A:w} {C:o}  type.
subtract/* : subtract (A * B * C) D.
```

It returns

```
subtract/* : {A:w} {C:w} {B:w} {D:o} subtract (B * A) D.
```

which is manifestly wrong. The dummy argument D is there because I'm checking for '@type' by detecting which type families have w as their last argument. I should figure out a less hacky way of doing that.

Ok, I had tl in a couple places while sorting world variables instead of tl'. Grr.

I notice that I may have to deal with fvars as well as bvars, which I don't know how to or attempt to sort during normalization. This doesn't *seem* to screw anything up, but it makes me nervous.

(**TW**) Edits.

(**TFU**) HLF implementation.

(**TFC**) HLF implementation.

(**TE**) Gave receipts to Jen. Need to: pay electric, gas, phone, balance checkbook

2009.1.9

Reading Hazelwinkel et al's paper titled "The Ubiquity of Coxeter-Dynkin Diagrams". Wonderfully straightforward writing. Two things I got out of it:

One.

A coxeter diagram is translated to a group by taking the free group on the vertices modulo $(ss')^{m(s,s')} = 1$ where $m(s, s')$ is the edge weight

between s and s', taken to be 1 for self-edges, 2 for missing edges, 3 for conventionally unlabelled edges, and m for edges labelled m.

One of the canonical lists of coxeter diagrams I see a lot in Baez's stuff classify *the coxeter groups that happen to be finite.*

Two.

The lie algebra of a lie group is exactly the collection of left-invariant vector fields on it, and is exactly the set of tangent vectors at the identity. LIVFs aren't as scary as I supposed.

Take a vector field to be a derivation X_U that (locally) takes a function $U \to \mathbb{R}$ to another $U \to \mathbb{R}$, for any open set U that fits in a chart. Of course the derivation has to be compatible with restriction in a presheafy sort of way, and it also has to satisfy the Leibniz law for multiplication. The vector field F that's secretly there shows up in the sense that $X_U(f)$ for $f : U \to \mathbb{R}$ is $\lambda x : U.(D_{F(x)}f)(x)$, where $D_{\vec{u}}$ is the derivative in the $\vec{u}$ direction.

To be left-invariant, X_U has to satisfy $X_{y^{-1}U}(f \circ \lambda_y) = X_U(f) \circ \lambda_y$ for any $y \in G$ where λ_y is the obvious function $G \to G$ that hits you with y on the left. The Lie algebra structure comes from the fact that if X and Y are LI, then so too is $[X, Y] = XY - YX$.

2009.1.10

Todo today:

- writing on LICS paper ✓
- Update TL ✓
- pay electric bill ✓
- write CV ✓
- implementation goal: get unification to postpone constraints, see what happens. ✓ It adds constraints if it finds evars at the head of either side of the equation, but then it finds a constraint not of that form.
- Get either tags or undirected links working in paraphrase

Paradiddle: 108bpm Seven-stroke roll: 125bpm.

I think I can do a 'native' cut elimination theorem for a hybrid version of linear logic including positive connectives if I have a different version of the cut principle for the two polarities.

Say

$$\begin{array}{rcl} \text{Positives } P & ::= & P \otimes P \mid P \oplus P \mid \downarrow N \\ \text{Negatives } N & ::= & P \multimap N \mid N \& N \mid \uparrow P \\ \text{Contexts } \Gamma & ::= & \cdot \mid \Gamma, \alpha, P[\alpha] \end{array}$$

I might nearly just as well in this setting identify world variables with term variables! But in the more general hybrid case I cannot. Nonetheless I live in the 'regular world' where each positive assumption in the context comes with its own worldvar.

The cut principles are

$$\frac{\Gamma; \Omega \vdash P[p] \qquad \Gamma, P[\alpha] \vdash J}{\Gamma; \Omega \vdash \{p/\alpha\}J} + \qquad \frac{\Gamma; \Omega \vdash N[\{p/\alpha\}q] \qquad \Gamma; \alpha, N[q] \vdash J}{\Gamma; \Omega \vdash \{p/\alpha\}J} -$$

Ω is optionally $\alpha, N[p]$. The judgment form when Ω is nonempty, i.e. when we're in left focus, is $\Gamma; \alpha, N[p] \vdash J$ where α is bound in p and J (and conceivably N in a full hybrid setting) but nowhere else. The left and right rules for the shifts are

$$\frac{\Gamma \vdash P[p]}{\Gamma \vdash \uparrow P[p]} \uparrow R \qquad \frac{\Gamma, \beta, P[\beta] \vdash J}{\Gamma; \alpha, \uparrow P[p] \vdash \{p/\beta\}J} \uparrow L$$

$$\frac{\Gamma \vdash N[p]}{\Gamma \vdash \downarrow N[p]} \downarrow R \qquad \frac{\Gamma; \beta, N[\beta] \vdash \{\beta/\alpha\}J}{\Gamma, \alpha, \downarrow N[\alpha] \vdash J} \downarrow L$$

Should check the LCD[1] cases here.

$$\frac{\dfrac{\Gamma; \beta, N[\beta] \vdash P[\{\beta/\gamma\}p]}{\Gamma, \downarrow N[\gamma] \vdash P[p]} \qquad \Gamma, P[\alpha] \vdash J}{\Gamma, \downarrow N[\gamma] \vdash \{p/\alpha\}J} + \qquad \frac{\dfrac{\Gamma; \beta, N[\beta] \vdash P[\{\beta/\gamma\}p] \qquad \Gamma, P[\alpha] \vdash J}{\Gamma; \beta, N[\beta] \vdash \{\{\beta/\gamma\}p/\alpha\}J}+}{\Gamma; \beta, N[\beta] \vdash \{\beta/\gamma\}\{p/\alpha\}J} = \qquad \Gamma, \downarrow N[\gamma] \vdash \{p/\alpha\}J$$

[1] I think it might be better to call what I had been calling LCL and LCR, instead LCD and LCE, by the usual naming convention of the two inputs to the cut principle being D and E, and not particularly left and right... Indeed I think Frank's (and perhaps many others') convention is that 'left commutative' cuts are those that operate on D. I'd rather use the word 'left' for 'pertains to sequent left rule.'

Here we're using that γ doesn't occur in J.

$$\dfrac{\dfrac{\Gamma;\beta,M[\beta]\vdash N[\{\beta/\gamma\}\{p/\alpha\}q]}{\Gamma,{\downarrow}M[\gamma]\vdash N[\{p/\alpha\}q]}\qquad \Gamma;\alpha,N[q]\vdash J}{\Gamma,{\downarrow}M[\gamma]\vdash\{p/\alpha\}J}\,-$$

Noting that $\{\beta/\gamma\}\{p/\alpha\}q = \{\{\beta/\gamma\}p/\alpha\}q$ we do

$$\dfrac{\Gamma;\beta,M[\beta]\vdash N[\{\{\beta/\gamma\}p/\alpha\}q]\qquad \Gamma;\alpha,N[q]\vdash J}{\dfrac{\Gamma;\beta,M[\beta]\vdash\{\{\beta/\gamma\}p/\alpha\}J}{\Gamma,{\downarrow}M[\gamma]\vdash\{p/\alpha\}J}}\,-$$

This doubling is getting tedious. Define a predicate (J,ω,p) with two rules (where ω is either of the form $\alpha,P[\alpha]$ or else the focus cell $\underline{\alpha,N[q]}$)

$$\dfrac{}{(P[p],P[\alpha],p)}\qquad\dfrac{}{(N[\{p/\alpha\}q],\underline{\alpha,N[q]},p)}$$

Then the cut principle is

$$\dfrac{\Gamma,\Omega\vdash J\qquad\Gamma,\omega\vdash J'\qquad(J,\omega,p)}{\Gamma,\Omega\vdash\{p/\alpha_\omega\}J'}$$

and left commutative D for $\downarrow$ is just

$$\dfrac{\dfrac{\Gamma,\underline{\beta,N[\beta]}\vdash\{\beta/\gamma\}J}{\Gamma,{\downarrow}N[\gamma]\vdash J}\qquad\Gamma,\omega\vdash J'\qquad(J,\omega,p)}{\Gamma,{\downarrow}N[\gamma]\vdash\{p/\alpha_\omega\}J'}$$

$$\mapsto\quad\dfrac{\Gamma,\underline{\beta,N[\beta]}\vdash\{\beta/\gamma\}J\qquad\Gamma,\omega\vdash J'\qquad(\{\beta/\gamma\}J,\omega,\{\beta/\gamma\}p)}{\dfrac{\Gamma,\underline{\beta,N[\beta]}\vdash\{\beta/\gamma\}\{p/\alpha_\omega\}J'}{\Gamma,{\downarrow}N[\gamma]\vdash\{p/\alpha_\omega\}J'}}$$

(noting that γ is not free in ω or J' and that the predicate $(,,)$ is stable under substitution) and left commutative D for $\uparrow$ is

$$\dfrac{\dfrac{\Gamma,\beta,P[\beta]\vdash J}{\Gamma,\underline{\gamma,{\uparrow}P[q]}\vdash\{q/\beta\}J}\qquad\Gamma,\omega\vdash J'\qquad(\{q/\beta\}J,\omega,p)}{\Gamma,\underline{\gamma,{\uparrow}P[q]}\vdash\{p/\alpha_\omega\}J'}$$

Here I seem slightly stuck because I don't know how to back out of this substitution...

2009.1.11

Todo today:

- whiteboard session — try demanding negative inversions to simplify cut ✓ This seems to work but then identity is hard as usual
- Update TL ✓
- more writing of CV — are references appropriate? Cover page? Research statement? Anything else? ✓ Looks like references can be easily mentioned separately. MPI wants one with a different affiliation from the submitter. I wonder who would be good for that?
- implementation goal: Find out why it's hard to add constraints on current example, add them back. It should be no mystery why abstraction is falling off the edge of the world if there are constraints remaining, right? ✓ Problem seemed to be that I wasn't accounting for projections from block variables. Now it's adding all constraints that it needs to, but still yields a few missing cases. It's quite possible that unification is failing very far away.
- writing on LICS paper
- pay gas bill
- Work more on getting tags to be useful. Is there any *easy* way to insinuate in some algebra on tags?

Paradiddle: (half = RlrrLrll) still somewhat comfortable at 116bpm, start to break down around 120bpm.

Seven-stroke roll: (whole = rlrlrlR.lrlrlrL.) Can pull off 130bpm, but there I start to lose precision.

Right three-stroke roll: (quarter = R..lrl) 120bpm

Left three-stroke roll: (quarter = L..rlr) 110bpm

French roll: (half = RrrLll) 120bpm

LCD↑ really does have apparent problems if I allow interleaving of negative asynchronous decomposition. I seem to always encounter an expression that I know to arise from substitution in two different ways, and untangling it seems hard.

2009.1.12

Here are some thoughts about how to use my 'focalizing linear logic in itself' ideas to do right inversion, too.

$\Uparrow P = \uparrow P$ and $\Uparrow N = N$
$(N \multimap N)^\bullet_L = N^\bullet_R \multimap \uparrow\downarrow N^\bullet_L$
$(N \multimap N)^\bullet_R = \downarrow N^\bullet_L \multimap N^\bullet_R$
$(N \multimap N)^\circ_L = \downarrow\uparrow N^\circ_R \multimap \uparrow\downarrow N^\circ_L$
$(N \multimap N)^\circ_R = \downarrow N^\circ_L \multimap \downarrow\uparrow N^\circ_R$

Lemma 0.42 *These are equivalent:*

- $\Gamma \vdash_i P$
- $(\downarrow\Uparrow\Gamma)^\bullet, q \vdash (\uparrow P)^\bullet$
- $(\downarrow\Uparrow\Gamma)^\circ, q \vdash (\uparrow P)^\circ$
- $\Gamma \vdash P$

and these are equivalent:

- $\Gamma \vdash_i N$
- $(\downarrow\Uparrow\Gamma)^\bullet \vdash N^\bullet$
- $(\downarrow\Uparrow\Gamma)^\circ, q \vdash \uparrow(N^\circ)$
- $\Gamma \vdash N$

Proof $\downarrow\uparrow N^\circ_R = N^\bullet_R$ and $\downarrow N^\circ_L = \downarrow N^\bullet_L$
$\downarrow\uparrow(\downarrow N^\circ_L \multimap \downarrow\uparrow N^\circ_R) = \downarrow N^\bullet_L \multimap N^\bullet_R$ ■

2009.1.13

Had several minor breakthroughs on the positives-in-HLF front.

$$
\begin{array}{rcl}
\text{Unrestricted } A & ::= & U_p N \\
\text{Positives } P & ::= & F_p A \mid \Sigma P[a].P \mid P \oplus P \mid 0 \\
\text{Negatives } N & ::= & \uparrow P \mid \Pi P[a].N \mid N \,\&\, N \mid \top \mid @_p N \mid \downarrow a.N \\
\text{Inv. Contexts } \Omega & ::= & \cdot \mid \Omega, P[a] \\
\text{Active Contexts } \Delta & ::= & \Omega \mid N[p] \\
\text{Contexts } \Gamma & ::= & \cdot \mid \Gamma, A \\
\text{Passive Concs } J & ::= & P[p] \\
\text{Concs } K & ::= & (;\Delta \vdash J) \mid (\vdash \underline{P[p]}) \mid (;\Omega \vdash N[p]) \mid (\vdash A)
\end{array}
$$

Judgments
ΓK

	Commutative	Principal
$+$	$\dfrac{\Gamma;\Delta \vdash P[p] \qquad \Gamma; P[a] \vdash J}{\Gamma;\Delta \vdash (p/a)J}$	$\dfrac{\Gamma \vdash \underline{P[p]} \qquad \Gamma;\Omega, P[a] \vdash J}{\Gamma;\Omega \vdash (p/a)J}$
$-$	$\dfrac{\Gamma \vdash A \qquad (\Gamma, A)K}{\Gamma K}$	$\dfrac{\Gamma;\Omega \vdash N[q] \qquad \Gamma; N[q] \vdash J}{\Gamma;\Omega \vdash J}$

The story of how these show up in proof search 'in the wild' is left-to-right, top-to-bottom. Commutative goes to (the same polarity) principal when it runs out of commutative stuff to do, and principal goes back to commutative (of the opposite polarity) when it encounters a shift.

Shifts

$$\frac{\Gamma;\cdot \vdash N[q]}{\Gamma \vdash U_q N} \qquad \frac{\Gamma; N[q] \vdash J}{\Gamma, U_q N;\cdot \vdash J}\, s$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \underline{F_p A[p]}}\, s \qquad \frac{\Gamma, A;\Omega \vdash (p/a)J}{\Gamma;\Omega, F_p A[a] \vdash J}$$

$$\frac{\Gamma;\Omega \vdash P[p]}{\Gamma;\Omega \vdash \uparrow P[p]} \qquad \frac{\Gamma; P[a] \vdash J}{\Gamma; \uparrow P[p] \vdash (p/a)J}\, s$$

$$t = U_* \top \qquad ! = F_\epsilon U_\epsilon$$

$$1_q = F_q t \qquad 1 = 1_\epsilon$$

$$P \otimes P' = \Sigma P[a].\Sigma P'[b].1_{a*b}$$

$$P \multimap N = \downarrow b.\Pi P[a].@_{a*b} N$$

$$\forall \alpha.N \equiv \Pi t[\alpha].N$$

$$\exists \alpha.P \equiv \Sigma t[\alpha].P$$

2009.1.14

Current version of sowreap:

```
%hlf.

nat : type. %name nat N.
s : nat -> nat.
z : nat.

plus : nat -> nat -> nat -> type.
plus/z : plus z N N.
plus/s : plus (s M) N (s P)
  <- plus M N P.

seed : @type.

reap : nat -> @type.
reap/z : reap z.
reap/s : seed -o reap N -o reap (s N).

sow : nat -> nat -> @type.
sow/z : reap M -o sow z M.
sow/s : (seed -o sow N M) -o sow (s N) M.


plus-lemma : plus A B C -> plus (s z) D B -> plus (s A) D C -> type.
plus-lemma2 : plus (s z) B C -> plus (s z) (s B) (s C) -> type.
plus-lemma3 : plus z B z -> plus A B C -> plus A z C -> type.

seed-subtract : seed @ A -> reap P @ (A * B)
      -> reap N @ B -> plus (s z) N P -> type.

seed-subtract/*/that : seed-subtract S (reap/s ^ H ^ TL) (reap/s ^ H ^ TL') PLUSOUT
   <- seed-subtract S TL TL' PLUS1
   <- plus-lemma2 PLUS1 PLUSOUT.

seed-subtract/*/this : seed-subtract S (reap/s ^ S ^ TL) TL (plus/s plus/z).

strengthen : ({a:w} {x:seed @ a} reap N @ P) -> reap N @ P -> type.

main-lemma : sow M P @ A -> reap N @ A -> plus M N P -> type.
main-lemma/s : main-lemma (sow/s ^ SOW) REAP PLUS
   <- ({a:w} {x:seed @ a} main-lemma (SOW a x) (REAP' a x) PLUS')
   <- ({a:w} {x:seed @ a} seed-subtract x (REAP' a x) (REAP'' a x) PLUS'')
   <- strengthen REAP'' REAP
   <- plus-lemma PLUS' PLUS'' PLUS.
main-lemma/z : main-lemma (sow/z ^ REAP) REAP plus/z.

reap-zlemma : reap N -> plus z N z -> type.

thm : sow M P -> plus M z P -> type.
thm/ : thm SOW PLUS''
<- main-lemma SOW REAP PLUS
<- reap-zlemma REAP PLUS'
<- plus-lemma3 PLUS PLUS' PLUS''.
```

Going to try to revise seed-subtract to do strengthening *en passant*.

It nearly worked — except at the very bottom it's highly problematic in that I want to split on a variable but even 'on paper' several cases are apparently rather intractable. I wonder if I introduced an equality predicate on worlds to reify some of the contsraints I could get away with it?

	$\bullet_L$	$\bullet_R$	$\circ_L$	$\circ_R$
$P \multimap N$	$P^\bullet_L \multimap N^\bullet_R$	$uP^\bullet_L \multimap N^\bullet_R$	$UFP^\circ_R \multimap FUN^\circ_L$	$UP^\circ_L \multimap UFN^\circ_R$
$P \otimes P$	$f(uP^\bullet_L \bullet uP^\bullet_L)$	$P^\bullet_R \bullet P^\bullet_R$	$F(UP^\circ_L \otimes UP^\circ_L)$	$UFP^\circ_R \otimes UFP^\circ_R$
$\uparrow P$	$ff^*uP^\bullet_L$	$uFP^\bullet_R$	FUP°_L	UFP°_R
$\downarrow N$	$fUN^\bullet_L$	$u^*N^\bullet_R$	FUN°_L	UFN°_R

$$U = !(<> \Rightarrow \text{—}) \qquad u = > \Rightarrow \text{—} \qquad u^* = < \Rightarrow \text{—}$$
$$F = (!\text{—})\bullet <> \qquad f = \text{—}\bullet > \qquad f^* = < \bullet\text{—}$$

$$FP^\bullet_R = FP^\circ_R$$
$$UN^\bullet_L = UN^\circ_L$$
$$u^* N^\bullet_R = N^\circ_R$$
$$fuP^\bullet_L = fUP^\circ_L$$

2009.1.15

Here is a funny variant of linear logic. The syntax is

Valid A	$::=$	UB
Linear B	$::=$	$FA \mid A \Rightarrow B \mid A \,!\!\otimes\, B$

The judgments are $\Delta \vdash_n B$ and $\Gamma \vdash_0 A$. Δ can be a mix of A and B but Γ is only A. As are subject to weakening, contraction, while Bs are not.

The cut principle is

$$\frac{\Delta_1 \vdash_{n_1} X \qquad \Delta_2, X \vdash_{n_2} \vdash Y}{\Delta_1, \Delta_2 \vdash_{n_1+n_2} Y}$$

The rules for U and F go like

$$\frac{\Gamma \vdash_1 B}{\Gamma \vdash_0 UB} \qquad \frac{\Delta, B \vdash_n B'}{\Delta, UB \vdash_{n+1} B'}$$
$$\frac{\Gamma \vdash_0 A}{\Gamma \vdash_1 FA} \qquad \frac{\Delta, A \vdash_{n+1} B}{\Delta, FA \vdash_n B}$$

And the other connectives

$$\frac{\Delta, A \vdash_n B}{\Delta \vdash_n A \Rightarrow B} \qquad \frac{\Gamma \vdash_0 A \qquad \Gamma, \Delta, B \vdash_n B'}{\Gamma, \Delta, A \Rightarrow B \vdash_n B'}$$

$$\frac{\Gamma \vdash_0 A \qquad \Gamma, \Delta \vdash_n B}{\Gamma, \Delta \vdash_n A \,!\!\otimes\, B} \qquad \frac{\Delta, A, B \vdash_n B'}{\Delta, A \,!\!\otimes\, B \vdash_n B'}$$

N^L and N^R are propositions.

X	X^L	X^R
$P \multimap N$	$P^R_{\multimap}(N^L)$	$P^L_{\multimap}(N^R)$
$\uparrow P$	$P^L_{\otimes}(q)$	$UFP^R_{\otimes}(1)$

P^L_f and P^R_f are functions from prop to prop, if f is a binary operator on props.

X	X^L_f	X^R_f
1	id	id
$P \otimes P$	$P^L_f \circ P^L_f$	$P^R_f \circ P^R_f$
$P \oplus P$	$f^{\oplus}(P^L_f, P^L_f)$	$f^{\oplus}(P^R_f, P^R_f)$
$\downarrow N$	$f^!(UN^L, —)$	$f(N^R, —)$

where $\multimap^! = \Rightarrow$ and $\otimes^! = !\otimes$. also $\multimap^{\oplus} = \&$ and $\otimes^{\oplus} = \oplus$.

The all-pause translation is the same as earlier:

X	$X^L_\circ$	$X^R_\circ$
$P \multimap N$	$UFP^R_\circ \multimap FUN^L_\circ$	$UP^L_\circ \multimap UFN^R_\circ$
$P \otimes P$	$F(UP^L_\circ \otimes UP^L_\circ)$	$UFP^R_\circ \otimes UFP^R_\circ$
$\uparrow P$	$FUP^L_\circ$	$UFP^R_\circ$
$\downarrow N$	$FUN^L_\circ$	$UFN^R_\circ$

I might conjecture that $UN^L_\circ = UN^L$. I would then need $UP^L_\circ = UP^L_\otimes(q)$. The round trip around the other shift works. To get lolli I could require $U(P^R_\circ \multimap B) = U(P^R_{\multimap} B)$. For the case of $P = \downarrow N$ I would need $U(UFN^R_\circ \multimap B) = U(N^R \multimap B)$ so it would suffice to have $FN^R_\circ = FN^R$. For the other shift I want $FP^R_\circ = FP^R_\otimes(1)$. For the right case of lolli I need $F(UP^L_\circ \multimap UFN^R_\circ) = FP^L_{\multimap}(N^R)$ and it would suffice that

$$F(UP^L_\otimes(q) \multimap UFN^R) = FP^L_{\multimap}(N^R)$$

$$\begin{array}{cc} UN^L_\circ = UN^L & FN^R_\circ = FN^R \\ UP^L_\circ = UP^L_\otimes(q) & FP^R_\circ = FP^R_\otimes(1) \\ F(UP^L_{\multimap}(q) \multimap UFX) = FP^L_{\multimap}(X) & U(P^R_\circ \multimap B) = UP^R_{\multimap}(B) \end{array}$$

$$F(UP^L_{\multimap}(q) \otimes UP^L_{\multimap}(q)) = FU(P^L_{\multimap}(P^L_{\multimap}(q)))$$

Arg this is a huge mess.

Let me do currying separately.

$$(P \multimap N)^* = P^{\multimap}(N^*)$$

$$(P \otimes P)^f = P^f \circ P^f$$

$$(\downarrow N)^f = f(\downarrow N^*, —)$$

$$(\uparrow P)^* = P^\bullet(\uparrow 1)$$

If $f : P \times N \to N$ then $P^f : N \to N$.

$$P \bullet N = \uparrow(P \otimes \downarrow N)$$

Here I want $N^* = N$, so I need $P \multimap N = P^{\multimap}(N)$ and $\uparrow P = P^\bullet(\uparrow 1)$.

For $P = \downarrow N$ this last gives $\uparrow\downarrow N = \downarrow N^* \bullet \uparrow 1 = \uparrow(\downarrow N^* \otimes \downarrow\uparrow 1)$. For $P = P_1 \otimes P_2$ it would be $\uparrow(P_1 \otimes P_2) = P_1^\bullet(P_2^\bullet(\uparrow 1)) = P_1^\bullet(\uparrow P_2)$.

So clearly I want the stronger result that $P^\bullet(N) = P \bullet N$. Okay, that seems to work, great.

$$(P \multimap N)^* = P^-(N^*)$$

$$(P \otimes P)^f = P^f \circ P^f$$

$$(\downarrow N)^+ = \uparrow(\downarrow N^* \otimes \downarrow —)$$

$$(\downarrow N)^- = \downarrow N^* \multimap —$$

$$(\uparrow P)^* = P^+(\uparrow 1)$$

$$(P \oplus P)^- = P^- \mathbin{\&} P^-$$

$$(P \oplus P)^+ = P^+ \oplus P^+$$

2009.1.17

Trying to avoid currying by doing destination-passing style instead. Think I might have to make the blur translation allow arbitrary tokens at many points.

X	$X^L_\bullet$	$X^R_\bullet$
$P \multimap N$	$P^R_\bullet \multimap N^L_\bullet$	$a \mapsto \forall b. U_b P^L_{\bullet a} \multimap N^R_{\bullet b}$
$P \otimes P$	$a \mapsto \exists b. U_b P^L_{\bullet a} \otimes P^L_{\bullet b}$	$UFP^R_\bullet \otimes UFP^R_\bullet$
$\uparrow P$	$P^L_{\bullet\star}$	$a \mapsto U_a F P^R_\bullet$
$\downarrow N$	$a \mapsto F_a U N^L_\bullet$	$N^R_{\bullet\star}$

X	$X^L_\circ$	$X^R_\circ$
$P \multimap N$	$UFP^R_\circ \multimap FUN^L_\circ$	$a \mapsto (ub.P^L_{\circ b}) \multimap U_a(fb.N^R_{\circ b})$
$P \otimes P$	$a \mapsto F_a(ub.P^L_{\circ b}) \otimes (ub.P^L_{\circ b})$	$UFP^R_\circ \otimes UFP^R_\circ$
$\uparrow P$	$P^L_{\circ\star}$	$a \mapsto U_a FP^R_\circ$
$\downarrow N$	$a \mapsto F_a UN^L_\circ$	$N^R_{\circ\star}$

	L	R
-	$UN^L_\bullet = UN^L_\circ$	$N^R_{\bullet a} = U_a(fb.N^R_{\circ b})$
+	$P^L_{\bullet a} = F_a(ub.P^L_{\circ b})$	$FP^R_\bullet = FP^R_\circ$

$$\forall b.U_b P^L_{\bullet a} \multimap N^R_{\bullet b} = U_a f\gamma.((ub.P^L_{\circ b}) \multimap U_\gamma(fb.N^R_{\circ b}))$$

$$\forall b.U_b F_a(ud.P^L_{\circ d}) \multimap U_b(fc.N^R_{\circ c}) =$$

$$U_a F_a(ud.P^L_{\circ d}) \multimap U_a(fc.N^R_{\circ c}) = ((ub.P^L_{\circ b}) \multimap U_a(fb.N^R_{\circ b}))$$

Should have $F_a P^L_{\circ 1} = P^L_{\circ a}$
Know $P^L_{\bullet a} = F_a(ub.P^L_{\circ b})$.
Thus $UP^L_{\bullet\star} = UF(ub.P^L_{\circ b})$.
Need $UP^L_{\circ\star} = UF(ub.P^L_{\circ b})$.
I.e. $UFP^L_{\circ 1} = UF(ub.F_b P^L_{\circ 1})$. But this is provable!
The general adjoint equations seem to be

$$F_a X = F_a(ub.F_b X)$$

$$U_a X = U_a(fb.U_b X)$$

2009.1.18

Eureka! I think I've got it without the blur translation being weird. The thing I struggled with for hours and hours since early this morning was trying to find an inductive invariant for the asynchronous case that was an *equality* between proofs. Such a thing would be nice, but I don't think it obtains; instead, two back-and-forth lemmas per polarity seems right.

Looking at the content of those lemmas — or rather the converses of them that do not hold, or aren't inductively strong enough — I see so clearly the wall I was bashing my head against.

X	$X^L_\bullet$	$X^R_\bullet$
$P \multimap N$	$P^R_\bullet \multimap N^L_\bullet$	$a \mapsto \forall b. U_b P^L_{\bullet a} \multimap N^R_{\bullet b}$
$P \otimes P$	$a \mapsto \exists b. U_b P^L_{\bullet a} \otimes P^L_{\bullet b}$	$UFP^R_\bullet \otimes UFP^R_\bullet$
$\uparrow P$	$P^L_{\bullet\star}$	$a \mapsto U_a F P^R_\bullet$
$\downarrow N$	$a \mapsto F_a U N^L_\bullet$	$N^R_{\bullet\star}$

X	$X^L_\circ$	$X^R_\circ$
$P \multimap N$	$UFP^R_\circ \multimap FUN^L_\circ$	$UP^L_\circ \multimap UFN^R_\circ$
$P \otimes P$	$UP^L_\circ \otimes FUP^L_\circ$	$UFP^R_\circ \otimes UFP^R_\circ$
$\uparrow P$	$P^L_\circ$	$P^R_\circ$
$\downarrow N$	$N^L_\circ$	$N^R_\circ$

	S	A_1	A_2
-	$UN^L_\bullet = UN^L_\circ$	$N^R_{\bullet\star} \vdash UFN^R_\circ$	$U_a F N^R_\circ \vdash N^R_{\bullet a}$
+	$FP^R_\bullet = FP^R_\circ$	$FUP^L_\circ \vdash P^L_{\bullet\star}$	$P^L_{\bullet a} \vdash F_a U P^L_\circ$

2009.1.19

I want to define a relation $a.A \leq a.B$ by something sort of like the conjunction of $[\star/a]A \vdash [\star/a]B$ and $B \vdash A$, and this has nice properties like $U_a FU \leq U_a$ and $F_a \leq F_a UF$, but I can't figure out the right thing to say about its commuting with quantifiers. Tried a couple of generalizations to contexts rather than one bound variable, but nothing seemed satisfying.

2009.1.20

Taught 312 today, was much harder than I expected.

2009.1.21

I noticed that Simpson-style modal logic is also amenable to a more Pfenning-Davies style treatment if one only 'compiles' it in the same way I've been compiling HLF down to LF plus one ACU type.

Suppose the Simpson-style system looks like

$$A ::= \Box A \mid \Diamond A \mid A \wedge A \mid A \Rightarrow A \mid a$$

and that this is realized by hybrid operators like

$$A ::= \forall x.A \mid \exists x.A \mid A@x \mid A \wedge A \mid A \Rightarrow A \mid a$$

Then all we need to is suppose wlog that our atoms are negative, push all the @s down to those atoms, call the default modal strength $\star$ and invent a bunch more for the domain of quantification, and add $U_{x \leq \star}$, and replace $a@p$ with $U_{p \leq \star} a$. Then when it comes time to focus on a negative atom, we'll throw away all the propositions (including atoms) that aren't at the right world, and only be satisfied if we have the same atom at the same world.

Or I could just doubly index the atoms! That's much simpler.

Any *accessibility* relation on the atoms is orthogonal to (or at least supervenient on) any relation of strength on same.

Managing inequalities over some domain might be as easy as treating them like little implications, if I assume reflexivity and transitivity. I wonder what happens with fewer demands than that?

2009.1.22

Proving soundness and completeness for deriving entailments of inequalities over atoms in terms of *linear* implication is fairly easy because then you don't get a bunch of junk left over in the context complicating your life.

2009.1.23

Fidgeted with the implementation some more. Got cut elimination to work for no connectives at all, and it nearly works on just top. The important missing piece of the puzzle there seems to be reasoning on equations like $a = X[\xi] * Y[\xi']$ where a is not in the range of either ξ or ξ', both patterns, perhaps? I think it might work even without them being patterns. Because the point is, there's *no way at all* of the rhs yielding the parameter a, so we can fail.

At present it's postponed as a constraint, so we can't see that the E case of the hyp rule covers when, for example, we have a principal use of $\top$R in the D slot — doing so requires seeing that the ambient block cases of hyps don't overlap with requiring that a given hyp being used has a *locally* defined world, shifted out of range of blocks' worldvars.

2009.1.24

At the meeting yesterday Frank nudged me back towards thinking about the ordered token-passing protocol where you leave $q \rightarrow p$ at the left of the

context. I tried it very briefly this morning and couldn't get it to work, but something about the way that destination passing *does* work gives me renewed hope. Too bad the problem of guessing inductive invariants for these problems is so underconstrained.

Which reminds me — I can do a two-sided translation of Frank and Deepak's labelled judgmental S4, but then I have to guess the cut principle. Contrast diamond, which has a much purer one-sided translation $\Diamond = \Box(— \multimap p) \multimap p$.

2009.1.25

I think it's appropriate to consider a hypersequent calculus to be nonintuitionistic even if it consists of many sequents each one of which is single-conclusioned, because the hypersequent as a whole still permits many conclusions. In such a system possessing $\Box$ as defined by Restall the other day in his talk, you can still prove $\vdash A \vee \neg\Box A$, by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(A \vdash A) \mid (\vdash)}}{(\vdash A) \mid (\Box A \vdash)}}{(\vdash A) \mid (\vdash \neg\Box A)}}{\vdash A \vee \neg\Box A}$$

so it fails the disjunction property in that sense.

What is a proposition? I want to say: it's a thing that might be true, (contrast the classical: a thing that is true or false) and for which we know very clearly and definitely the conditions under which we would call it true.

This sounds like the definition of a positive connective according to the strain of thinking Noam subscribes to, and finally I get a feel for what he means by *defined by introduction*.

Then again I could say that I mean to apply this intuition to the whole of a sequent, in which case it would still sort of be on the positive side of the coin, since I'm talking about an *inductively* constructed proof-tree.

But there's a funny thing about the game of proof-checking that seems to fall back to being classical: I actually do want the proof-checking process to be decidable. A proof is a thing that *is or is not valid*, and provability is the constructive projection over all proofs; the thing is proven once a valid proof comes along, and I make no immediate definition of what unprovability means, and presumably all I admit to believing in is unprovability meaning that one can positively refute an assumption of provability.

But there I'm getting into hypotheses already! I wish to separate that out. For what does it mean to *hypothesize* something that we would oth-

erwise *prove*? Should my heuristic that different judgments typically have different connectives extend to the notion that assume-true and prove-true are actually different judgments?

Note that even if this were true, proving cut-elimination for one connective (or in this other world-view, proving that two connectives are 'partnered' appropriately) depends on commutative cases for all the rest — so it is still a slightly non-trivial thing to 'be a valid left connective' or 'be a valid right connective' to fit correctly into the larger picture.

Nagging questions:

- Why Cut and Identity?
- Why do arrows in a category have two endpoints, i.e. domain and codomain, and not more?

2009.1.26

Writing should eliminate unnecessary repetition.

2009.1.27

I cannot say: under any future circumstance, if asked question Q, I will answer with answer A, because I could imagine a future where I understand the words of Q and A differently. I want to speak of *the underlying question* Q but I don't know how.

2009.1.28

Remembered some scraps of thinking about thermo I did years ago. Suppose one system just has one state for each energy $n \in \mathbb{N}$. Then a collection of k such systems has call it $f(k,n) = \binom{k+n-1}{k-1} = \binom{k+n-1}{n}$ ways of containing energy n. If I take some other system under scrutiny that has say s_n states for each energy n, then the non-normalized probability I find it in some particular state of energy n when the total energy in the world is N is $f(k, N-n) = \binom{k+N-n-1}{N-n}$.

Further suppose $N = ak$ for some average enery a and let k go to ∞. Does anything coherent happen? I'm staring at

$$\binom{k+ak-n-1}{ak-n} = \frac{(k(a+1)-n-1)!}{(ak-n)!(k-1)!}$$

So let me first try to remember how approximate a general binomial coefficient with Stirling.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \approx \left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k} \sqrt{\frac{n}{2\pi k(n-k)}}$$

Substitute $b = a + 1$ and and we want

$$\binom{bk - n - 1}{bk - n - k}$$

substitute $\ell = bk - n - 1$ and we want

$$\binom{\ell}{\ell - (\ell + n + 1)/b}$$

2009.1.29

Neel suggested doing the syntactic focusing proof translation relationally — I thought at first it might obviate the asymmetry in the asynchronous reasoning, but then I couldn't get it to work after all.

2009.1.30

Some chords I enjoyed:

```
E C G D E C G B E C G D F C Bm B
A E G D F C Eb E A E G D F C B B
```

Talked to Chris a bit after the LF meeting about the 'array logic' business. She mentioned that the last thing she had thought of under that name was an ordered arrow that inserted a proposition for instance just to the left of the right of the context.

This made me think of generalizing the *number* of how much space a proposition counted as to something more like a set or list or interval or other structure of what its *extent* was supposed to be. For by recording the size, you're capturing that memory can't grow or shriink, but not that it can't move around.

The 'right but one' arrow I think fundamentally can't work, though; for it doesn't make any sense for an asychronous operation to insert something between two items that might be required to be adjacent for the proof to go through.

However, I could picture a negative modality that as a hypothesis means "true anywhere to the left (resp. right) of here". With a bit of polymorphism it'd be like $LA = \forall\alpha.\alpha \rightarrowtail A \bullet \alpha$ (resp. $RA = \forall\alpha.\alpha \twoheadrightarrow \alpha \bullet A$) The synthesized left and right rules would loook like

$$\frac{\alpha, \Xi \vdash A \ \mathsf{right}_\alpha}{\Xi \vdash LA} \qquad \frac{\Omega, A, \Gamma, \Delta \vdash C}{\Omega, \Gamma, LA, \Delta \vdash C}$$

plus a 'structural rule'

$$\frac{\Xi \vdash A}{\Xi, \alpha \vdash A \ \mathsf{right}_\alpha}$$

This is a little sketchy, since I'm focally 'grabbing' stuff that may not be positive atoms, but let's see if it works anyway. If I want to get the principal case working then the cut principle I guess might be

$$\frac{\Xi(\alpha) \vdash A \ \mathsf{right}_\alpha \qquad \Omega, A, \Gamma, \Delta \vdash C}{\Omega, \Xi(\Gamma), \Delta \vdash C}$$

which does case analysis only on the $\mathcal{D}$ slot, waiting for the rule to become the structural one. When it gets there, we get

$$\frac{\dfrac{\Xi \vdash A}{\Xi, \alpha \vdash A \ \mathsf{right}_\alpha} \qquad \Omega, A, \Gamma, \Delta \vdash C}{\Omega, \Xi, \Gamma, \Delta \vdash C}$$

which transforms easily.

I sit and think about interactions between the rules $\rightarrow L$ and LL and they seem nasty, but I can't seem to find a commutative case that acutally fails.

Alternatively I could imagine 'left-mobile' and 'right-mobile' judgments, which would break down the idea that you have separate zones, becuase they would still require a *place* in the judgment. To boot they would be examples of *directed* structural properties (like weakening, unlike contraction and exchange).

For we design our logical systems to be snappy ways of doing what would amount to taking all the ways that we could prove something by applying weakening or contraction or exchange step-by-step but then imposing proof irrelevance at the end of the day: for weakening we need only add contexts at the init rule, and at $\Box R$.

2009.1.31

Tried running thesis code on additive conjunction. Still doesn't work, spews lots of constraints out.

2009.2.1

I think of the three coercions between negative and positive and unrestricted hypotheses in HLF, two are actually $\downarrow$ and @.

2009.2.2

Here's an idea. Say the judgment form for a focusing system has many ordered contexts for asychronous positive work, and one may choose any right-end proposition from one of these to work on. The set of contexts is itself ordered, so that implication right can unambiguously toss its hypothesis on the rightmost.

Perhaps I can write something like $\Omega/\Omega/\Omega \cdots \Omega$ or more explicitly

$$\Omega ::= \cdot \mid \Omega/ \mid \Omega A$$

and have a structural rule that lets me erase an empty ordered context:

$$\frac{\Gamma; \Omega \vdash P}{\Gamma; \Omega/ \vdash P}$$

and implication right is like

$$\frac{\Gamma; \Omega, P \vdash N}{\Gamma; \Omega \vdash P \Rightarrow N}$$

and tensor left might be like

$$\frac{\Gamma; \Omega P_1, P_2/\Omega' \vdash J}{\Gamma; \Omega P_1 \otimes P_2/\Omega' \vdash J}$$

So here's a stab at the whole system. Structural rules:

$$\frac{\Gamma; [N] \vdash J}{\Gamma, N \vdash J} \qquad \frac{\Gamma \vdash [P]}{\Gamma \vdash P} \qquad \frac{\Gamma; \Omega \vdash J}{\Gamma; /\Omega \vdash J}$$

Shift rules:

$$\frac{\Gamma \vdash N}{\Gamma \vdash [\downarrow N]} \qquad \frac{\Gamma; \Omega/ \vdash P}{\Gamma; \Omega \vdash \uparrow P} \qquad \frac{\Gamma, N; \Omega/\Omega' \vdash J}{\Gamma; \Omega, \downarrow N/\Omega' \vdash J} \qquad \frac{\Gamma; P/ \vdash J}{\Gamma; [\uparrow P] \vdash J}$$

Cut principles:

$$\frac{\Gamma; \Omega_0 \vdash P \qquad \Gamma; \Omega P/ \vdash J}{\Gamma; \Omega/\Omega_0 \vdash J} \qquad \frac{\Gamma; \Omega_0 \vdash N \qquad \Gamma, N; \Omega \vdash K}{\Gamma; \Omega\Omega_0 \vdash K}$$

$$\frac{\Gamma \vdash [P] \qquad \Gamma; \Omega P/ \vdash J}{\Gamma; \Omega/ \vdash J} \qquad \frac{\Gamma; \Omega \vdash N \qquad \Gamma; [N] \vdash J}{\Gamma; \Omega/ \vdash J}$$

Principal

$$\frac{\dfrac{\Gamma; \Omega/ \vdash P}{\Gamma; \Omega \vdash \uparrow P} \quad \dfrac{\Gamma; P/ \vdash J}{\Gamma; [\uparrow P] \vdash J}}{\Gamma; \Omega/ \vdash J}$$

$$\frac{\dfrac{\Gamma \vdash N}{\Gamma \vdash [\downarrow N]} \quad \dfrac{\Gamma, N; \Omega/ \vdash J}{\Gamma; \Omega\downarrow N/ \vdash J}}{\Gamma; \Omega/ \vdash J}$$

This is hard to continue with since I keep reflexively simplifying the cut principles back down to what's strictly necessary to get the theorem to go through. But what I really want is a slightly more general form to get identity, since there I want to cut in things 'in the middle of phases'.

Back to positives in HLF, my current set of ideas is

$$\begin{array}{rcl} \text{True } A & ::= & @_p N \\ \text{Negative } N & ::= & \uparrow P \mid P \multimap N \mid \Pi x : A.N \\ \text{Positive } P & ::= & \downarrow \alpha.A \mid P \otimes P \end{array}$$

with some rules and cut principles

$$\frac{\Gamma \vdash P[p]}{\Gamma \vdash \uparrow P[p]} \qquad \frac{\Gamma; P[\alpha] \vdash J[q]}{\Gamma; \uparrow P[p] \vdash J[(p/\alpha)q]} \qquad \frac{\Gamma \vdash \underline{P[p]} \qquad \Gamma; \Omega, P[\alpha] \vdash J}{\Gamma \vdash (p/\alpha)J}$$

$$\frac{\Gamma \vdash N[p]}{\Gamma \vdash @_p N} \qquad \frac{\Gamma; N[p] \vdash J}{\Gamma, @_p N \vdash J} \qquad \frac{\Gamma \vdash N[p] \qquad \Gamma; N[p] \vdash J}{\Gamma \vdash J}$$

$$\frac{\Gamma \vdash (p/\beta)A}{\Gamma \vdash (\downarrow \beta.A)[p]} \qquad \frac{\Gamma, \alpha : \mathsf{w}, (\alpha/\beta)A; \Omega \vdash J}{\Gamma; \Omega, (\downarrow \beta.A)[\alpha] \vdash J}$$

2009.2.3

Why don't I have a nice collection of webpages, one per 'paper'? This would allow citations forward in time. Perhaps merely practical issues like LaTeX in HTML.

So if you include $\top$, sized ordered logic is genuinely different from orered logic. Consider trying to prove

$$\top^1 \bullet B^1 \bullet \top^2 \vdash \top^2 \bullet B^1 \bullet \top^1$$

You can't, but you can if you erase sizes.

Now on the other hand, for any proof in ordered logic, *there is some assignment of sizes* to the final sequent that respects a given assignment of sizes to atomic propositions (simply push sizes up through the derivation) but you're not guaranteed that it's the same sizing as the one you might have been separately handed for the last sequent *if top is involved*. If not, then I think the logic's conservative over ordered logic.

Reading Tsukada and Igarashi's TLCA '09 submission. Great stuff. It's the paper I wanted Taha and Nielsen '03 ("Environment Classifiers") to be.

The only thing I notice different between staged computation and full-on macro systems is the ability of macros to intensionally *analyze* code. But perhaps the moral of the story is, if you want to pick stuff apart, use your own damn positive type to build up a surrogate for the code, and transfer it over to the negative type of actual compiled code by using staging features of the language.

Still trying to figure out what's realling going on with 'cross-stage persistence', confusingly (to me) abbreviated CPS because I keep thinking of continuation-passing style. It seems to be some sort of basic monotonicity property.

2009.2.4

Looking at James Cheney's NLF work. I like it a lot. The automatic weakening of names means I really must confront encodings of affine things. It's possibile here that I might get away with it even in relevant HLF if all functions are additive enough to distribute the context of names around everywhere. I note that he culls names upon concretion, but he doesn't split these effectively 'linear' contexts at (ordinary) function application.

2009.2.5

I had this thought that there must be a common generalization of Levy's CBPV stuff and the Tskukada and Igarashi work on making Walid Taha's environment classifiers actually a logic. Damned if I can figure out what it is, though.

The Tsukada-Igarashi $\lambda^{\triangleright}$ system, as much as it makes sense as a logic, is really confounding me with why its operational system should work the way it does. For one thing, you always compute underneath $\Lambda\alpha$. For another, whether you compute under a λx depends on which level you're at.

I'm tempted to think this means that these are positive somehow, because despite my best efforts apparently I have been successfully brainwashed into thinking that values are symptoms of positive focus.

Two tangential thoughts on this:

One, why is it positive focus, really? Since in ML we ordinarily talk about closed evaluation, we need not focus on the right; it's just that the only stuff we can do is on the right, *until* we come to a lambda. On the other hand, we're supposed to stop if we get to a lazy pair, too. So maybe that's convincing.

Two, I can sort of begin to imagine a implication with polarity signature $+ \to + \to +$ as a macro, inasmuch as I could say (for all positive propositions)

$$(A \oplus B) \Rightarrow C \cong (A \Rightarrow C) \otimes (B \Rightarrow C)$$

$$(A \otimes B) \Rightarrow C \cong A \Rightarrow (B \Rightarrow C)$$
$$1 \Rightarrow C \cong C$$
$$0 \Rightarrow C \cong 1$$

but then it's much less clear what to do at atoms and at $\downarrow$, especially if I want things too be 'compatible with substitution of positive propositions or positive atoms' whatever that means. I could start by thinking of some more 'exponential identities' like

$$N \Rightarrow (A \otimes B) \cong (N \Rightarrow A) \otimes (N \Rightarrow B)$$
$$N \Rightarrow 1 \cong 1$$

but that only gets me so far, and besides, these aren't valid in linear logic thinking of $\Rightarrow$ as $\multimap$, whereas the other group is. Actually, not true for $(A \oplus B) \Rightarrow C \cong (A \Rightarrow C) \otimes (B \Rightarrow C)$. I would have wanted an ampersand there, and likewise really I have $0 \Rightarrow C \cong \top$ and not 1.

I suspect what Noam/Bob/Dan's HOF system would yield is effectively

$$N \Rightarrow P \cong \downarrow(\downarrow N \multimap \uparrow P)$$
$$a^+ \Rightarrow P \cong \downarrow(a^+ \multimap \uparrow P)$$

2009.2.6

It's perplexing how to realize totality for the decision procedure for equality of names

```
eq : name -0 name -0 bool -> @type.
eq/tt : {N :^ name} eq ^ N ^ N tt @ P.
eq/ff : {M :^ name} {N :^ name} eq ^ M ^ N ff @ P.
```

because it seems to depend on the invariant that the query is at a world that consists of one unique copy of each world variable in the context.

2009.2.7

Maybe, however, I can define another predicate

```
in : name -0 @type.
in/ : {N :^ name} in ^ N @ P.
```

and conclude a conditional effectiveness lemma

```
lemma : in ^ N @ P -> in ^ M @ P
   -> {B : bool} eq ^ M ^ N bool @ P -> type.
%mode lemma +INN +INM -BOOL -EQ
```

presumably with the two clauses

```
lemma/tt : lemma in/ in/ tt eq/tt
lemma/ff : lemma in/ in/ ff eq/ff
```

2009.2.8

Bad philosophy is atheist theology.