Notes

Jason Reed

March 15, 2007 – March 14, 2008

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"

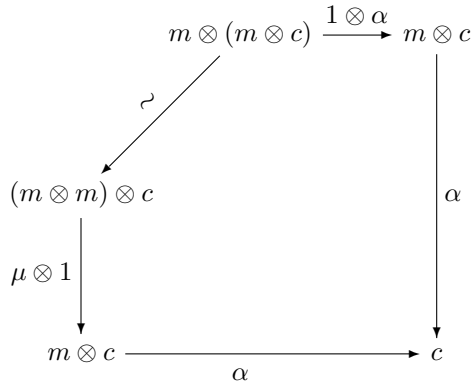
- "Looping" is a name for the process of getting a $k+1$-tuply monoidal $n-1$-category from a k-tuply monoidal n-category by choosing an object of it and index-shifting, getting a new operation.
- The coherence laws for braided (and perhaps plain) monoidal categories *don't* imply that every isomorphism 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 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.