Workshop on implementing Kosta Dosen's categorial programming and sheaves: Cut-elimination in the double category of profunctors with J-rule-eliminated adjunctions.

Short: There is now sufficient evidence (ref [6], [7]) that Kosta Dosen's ideas and techniques (ref [1], [2], [3], [4], [5]) could be implemented for proof-assistants, sheaves and applications; in particular cut-elimination, rewriting and confluence for various enriched, internal, indexed or double categories with adjunctions, monads, negation, quantifiers or additive biproducts; quantitative/quantum linear algebra semantics; presheaf/profunctor semantics; inductive-sheafification and sheaf semantics; sheaf cohomology and duality... It was difficult to discover the correct grammatical-formulation out of a dozen semantically-meaningful ones, but this ongoing implementation (ref [6]) should work in the next months because it can already be expressed (computationally) that right adjoint preserves weighted limits and that covering sieves are merely dependent-profunctors-types.

The problem of "formulations of adjunction", the problem of "unit objects" and also the problem of "contextual composition/cut" can be understood as the same problem. The question arises when one attempts to write precisely the counit/eval transformation ϵ_X ： catA(F G X, X) where F : catB ⟶ catA is the left-adjoint. One could instead write ϵ_1 ： catA[F ,1](G , 1) where the profunctor object/datatype catA[F ,1] is used in lieu of the unit hom-profunctor catA; in other words: F is now some implicit context, and the contextual composition/cut (in contravariant action) of some g : catB[1, G] (Z, Y) in the unit profunctor against ϵ_1 should produce an element of the same datatype: ϵ_1◦g : catA[F,1](Z,Y)... Ultimately Dosen-Petric (ref [5]) would be extended in this setting where some dagger compact closed double category, of left-adjoint profunctors across Cauchy-complete categories, is both inner and outer dagger compact closed where the dagger operation on profunctors (as 1-cells) coincides with the negation operation on profunctors (as 0-cells), optionally with sheaf semantics and cohomology.

The initial key insight of Dosen-Petric, about the cut-elimination formulation in the domain-specific language of categorial adjunctions, can be understood as a problem of "unit profunctor" in the double category of profunctors and the (inner) cut elimination lemma becomes synonymous with elimination of the "J-rule"; and recall that such equality/path-induction J-rule would remain stuck in (non-domain-specific) directed (homotopy) type theory.

Now the many "formulations of adjunctions" are for different purposes. Indeed the outer framework (closed monoidal category, with conjunction bifunctor $\wedge$ with right adjoint implication connective $\Rightarrow$) hosting such inner domain-specific language (unit-counit formulation of adjunctions) could itself be in another new formulation of adjunctions (lambda/eval bijection of hom-sets) where the implication bifunctor (with also contravariant argument _⇒_) is accumulated during computation via dinaturality (in contrast to the traditional Kelly-MacLane formulation).

For reference (§ 4.1.5 in [1]), Kosta Dosen says that an adjunction with left adjoint functor $F: catB \to catA$, right adjoint functor $G: catA \to catB$, counit transformation $\phi_A: F\ G\ A \to A$, and unit transformation $\gamma_B: B \to G\ F\ B$ is formulated as rewrite rules from any redex outer cut on the left-side to the contractum containing some smaller inner cut, where f_1, g_1 are the (fixed) parameters and f_2, g_2 are the natural variables (those naturality equations can be formulated generally for any transformation):

$$f_2 \circ (f_1)\text{“}1 \circ \phi \circ F\text{”} = (f_2 \circ f_1)\text{“}1 \circ \phi \circ F\text{”}$$

$$(f_1)\text{“}1 \circ \phi \circ F\text{”} \circ Gf_2 = (f_1 \circ f_2)\text{“}1 \circ \phi \circ F\text{”}$$

$$\text{“}1 \circ \phi \circ F\text{”}(g_1)\text{“} \circ F\text{”} \circ g_2 = \text{“}1 \circ \phi \circ F\text{”}(g_1 \circ g_2)$$

$$f_2 \circ \text{“}1 \circ \phi \circ F\text{”}(g_1) = \text{“}1 \circ \phi \circ F\text{”}(Gf_2 \circ g_1)$$

together with conversion (or rewrite) rules:

$$\text{“}1 \circ \phi \circ F\text{”}((f)\text{“}G \circ \gamma \circ 1\text{”}) = f$$

$$\text{“}1 \circ \phi \circ F\text{”}(\text{“}G \circ \gamma \circ 1\text{”}g) = Fg$$

where indeed the functions on arrows $F\ -$ and $G\ -$ of those functors are not primitive but are themselves the "consequential transformation" formulations "$1 \circ F\ -$" of the identity natural transformation…

New functorial lambda calculus. Now such Dosen-style technique may be specialized to the instance of closed monoidal categories where the conjunction bifunctor _∧_ has right adjoint implication bifunctor _⇒_ via lambda/eval bijection of hom-sets. Then dinaturality is used to accumulate the argument-component of the eval operation instead on its function-component:

$$\text{“}\epsilon_{B,O} \circ B \wedge (g)\text{”} \circ (x \wedge f)$$

$$= \text{“}\epsilon_{A,O} \circ A \wedge \big((x \Rightarrow O) \circ (g \circ f)\big)\text{”}, \qquad x: A \to B$$

where this evaluation rule uses implication bifunctors skewed by reindexing (the domain is left extended):

```
| Eval_cov_transf : Π [A B C X A' X' : Cat] [P : mod A B] [Q :
mod C X] [O : mod A' X'] [K : func B C] [F : func A A'] [L : func
X X'],
 P ⊢ F ◦> (Imply_cov_mod (O <◦  L) Q) <◦  K ⟶
(Tensor_cov_mod P (K ◦> Q)) ⊢ F ◦> O <◦  L
```

In fact, such antecedental/consequential transformations may be formulated systematically, similarly as the "J-rule" for equality/paths in (homotopy) type theory. *And most importantly, because the general "J-rule" hide some cuts, therefore cut-elimination signify that the general "J-rule" must also be eliminated/admissible/computational.* The J-rule constructor Unit_con_transf has this form, then the proof that right adjoint preserves weighted limits:

```
inductive cat : TYPE ≔  with func : Π (A B : cat), TYPE ≔
with mod : Π (A B : cat), TYPE ≔  with hom: Π [I A B : cat],
func I A ⟶ mod A B ⟶ func I B ⟶ TYPE ≔
with transf: Π [A' B' A B: cat], mod A' B' ⟶ func A' A ⟶ mod
A B ⟶ func B' B ⟶ TYPE ≔
| Unit_con_transf : Π [I A B J : cat] [F : func I A] [R : mod A B]
[G : func I B], Π (M : func J A), hom F R G ⟶ transf (Unit_mod
M F) Id_func (M ◦>> R) G ;

type λ [B J J' A I : cat] (F : func J B) (W : mod J' J) (F_⇇_W :
func J' B) (isl : isLimit F W F_⇇_W) (R : func B A) (L : func A B)
(isa : adj L R) (M : func I A),
((((M)_'◦> (Adj_cov_hom isa F)) ⇇2 (Id_transf W))
  ''◦ (limit_intro_transf isl (M ◦> L)))
    ''◦ ((Adj_con_hom isa M) ◦>'_(F_⇇_W)) ;
// : transf ((Unit_mod M (R <◦ F)) ⇇ W) Id_func (Unit_mod M
(R <◦ F_⇇_W)) Id_func
```

New grammatical topology. A sheaf is data defined over some topology, and sheaf cohomology is linear algebra with data defined over some topology. A closer inspection reveals that there is some intermediate formulation which is computationally-better that Cech cohomology: at least for the standard simplexes (line, triangle, etc.), then intersections of opens could be internalized as primitive/generating opens for the cover and become points in the nerve of this cover (as suggested by the barycentric subdivision). This redundant storage space for functions defined over the topology is what allows possibly-incompatible functions to be glued, and to prove the acyclicity for the standard simplex (and to compute how this acyclicity fails in the presence of holes in the nerve). For example, this sheaf data type, gives the gluing operation:

$$F(U0) \coloneqq \text{sum over the slice U0 U01} = \mathbb{Z} \oplus \mathbb{Z};$$

$$F(U1) \coloneqq \mathbb{Z} \oplus \mathbb{Z}; F(U01) \coloneqq \mathbb{Z}$$

$$F(U) = \text{kan extension} = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z};$$

$$\text{gluing: } F(U0) \oplus F(U1) \oplus F(U01) \to F(U)$$

$$\big((f0, f01), (g1, g01), (h01)\big) \mapsto (f0, g1, f01 + g01 - h01)$$

where the signed sum generalizes to higher degrees because the Euler characteristic is 1 (or inclusion–exclusion principle).

The implementation uses some local modifier $j\colon \Omega \to \Omega$ where $\Omega(A)$ is the classifier of (sub-)objects (sieves) of the object A, and where $j_A(\mathcal{U})(f) \coloneqq$ "$f^*\mathcal{U} \in J(X)$" is the (opaque) set of witnesses that the pullback-sieve $f^*\mathcal{U}$ is covering (remember that the truthness that $\mathcal{U}$ is covering is expressed as $\mathcal{U} \in J(A)$, iff $\forall f : some\ cover. j_A(\mathcal{U})(f)$), iff $j_A(j_A(\mathcal{U}))$.

Now dependent types are fibrations of (the double-category cograph of) (enriched) profunctors, where the vertical fibration is (co)cartesian. In fact, a sieve is literally defined as such (double-category codomain) fibration over some hom-profunctor, instead of defined as some subobject. Then the local modifier topology j becomes some predicate type with constructors for each topology axiom. Now the Gluing operation on data is formulated such that the codomain of the output is the sheafification modality and the input is any possibly-incompatible family of data which are indexed by the base-arrows of some covering sieve fibration...

New executable applications. Fibrations of (the double-category cograph) of (enriched) profunctors and their cut-elimination have immediate executable/computational applications to graphs transformations understood as categorial rewriting, where the objects are graphs (or sheaves in a topos), the vertical monomorphisms are pattern-matching subterms inside contexts and the horizontal arrows are congruent/contextual rewriting steps which have compositional/functorial effect with concurrency and associativity properties.

References:

[1] Dosen-Petric: Cut Elimination in Categories 1999;
[2] Proof-Theoretical Coherence 2004;
[3] Proof-Net Categories 2005;
[4] Coherence in Linear Predicate Logic 2007;
[5] Coherence for closed categories with biproducts 2022
[6] Cut-elimination in the double category of profunctors with J-rule-eliminated adjunctions: https://github.com/1337777/cartier/blob/master/cartierSolution12.lp
[7] Pierre Cartier