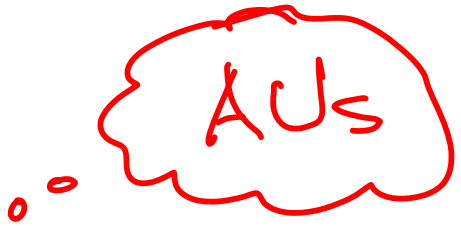


Arithmetic Universes and Type Theory

Steve Vickers
School of Computer Science
University of Birmingham

Sketches for AUs
"Contexts" for control over strictness

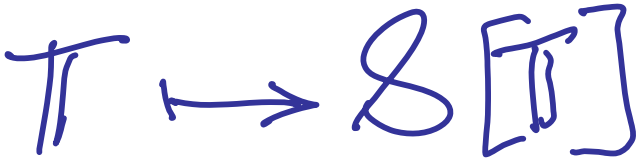
Questions:
(1) Using type theory to make AUs
* Reconcile Maietti's syntactic category with my sketches
(2) Using type theory to *compare* AUs
* "Dependent type theory of (generalized) spaces"

Padova Feb 2017

Why AUs?

Grothendieck says toposes are generalized spaces.

Each geometric theory has a classifying topos, somehow its "space of models".



First must choose a base elementary topos S "of sets" -
* determines what infinite disjunctions are allowed in geometric logic,
* used in constructing classifying topos.

If disjunctions are countable, will work for any S with nno
- but different S's give different classifying toposes.

Idea Get a classifying AU independent of S.

AU-functors give geometric morphisms,
hence "continuous maps" between the spaces.

Making AUs:

Type theory

Syntactic category -
Types and terms for objects and morphisms.
Type constructors for pullbacks, list objects etc.
Term constructor for associated morphisms.

Maietti:
Modular correspondence ...

Maietti:
Joyal's AUs via TT

Universal algebra

Present by generators (objects, morphisms) and relations.
Operators for pullbacks, list objects etc, and associated morphisms.

Palmgren, Vickers:
Partial Horn logic and cartesian categories

Type theory

Universal algebra

Maietti:
Modular correspondence ...

Maietti:
Joyal's AUs via TT

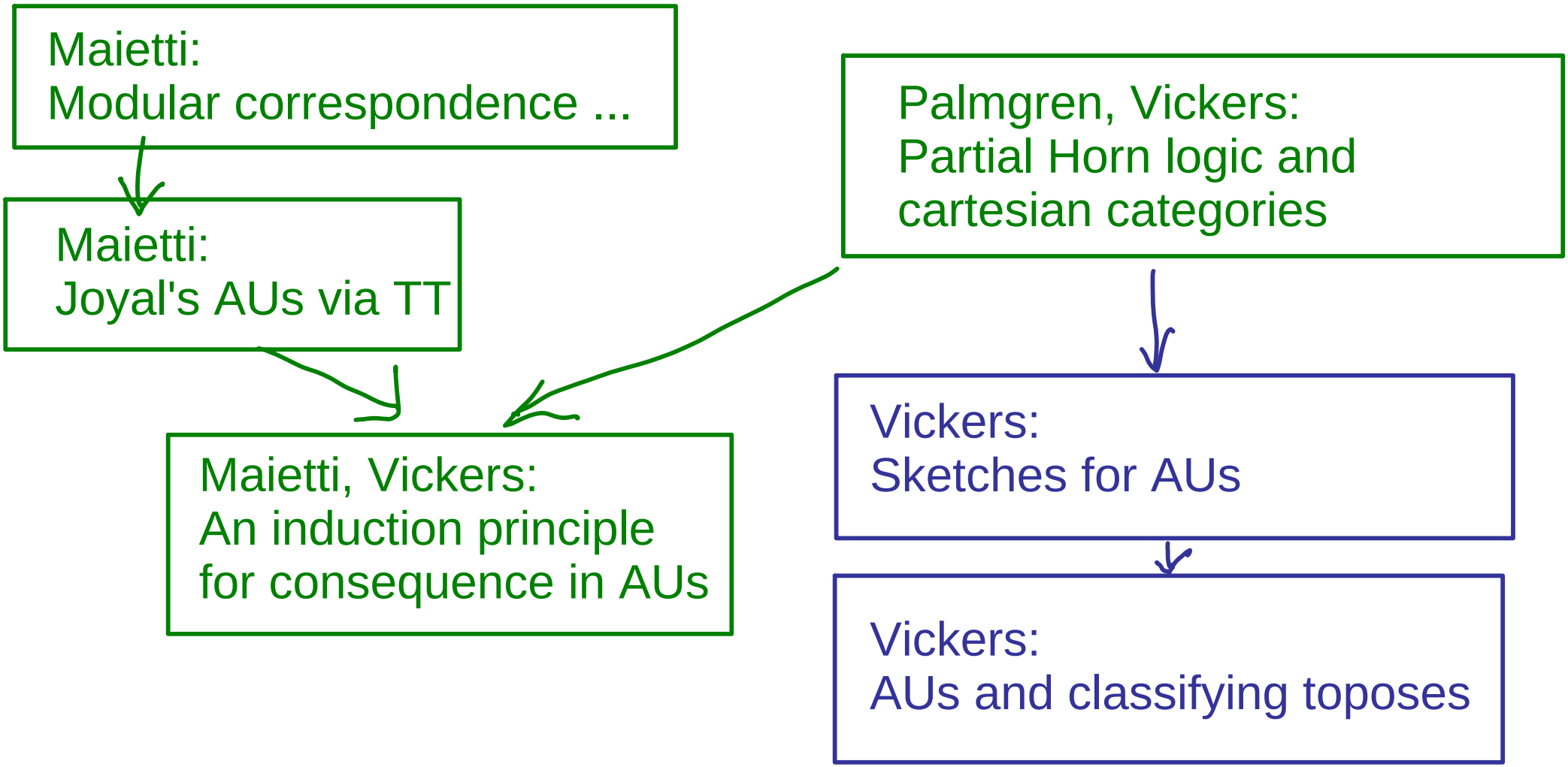
Palmgren, Vickers:
Partial Horn logic and
cartesian categories

Maietti, Vickers:
An induction principle
for consequence in AUs

Hope: Exploit TT for calculations in AU presented by generators and relations.
Didn't work out -
?? Universal characterization of syntactic category.
?? Issues of strictness for AU-functors.

Type theory

Universal algebra



Sketches

A directed graph (nodes, edges) names *all* objects and morphisms involved in a model.
Commutativities specify that diagrams commute.
Universals specify particular properties.

e.g. set M with binary operation u:

Product universal specifies (N, p1, p2) is a product cone

Models of a sketch

Interpret

nodes

edges

as

objects

morphisms

respecting commutativities and universals.

Strictly or non-strictly? e.g. for product universals:

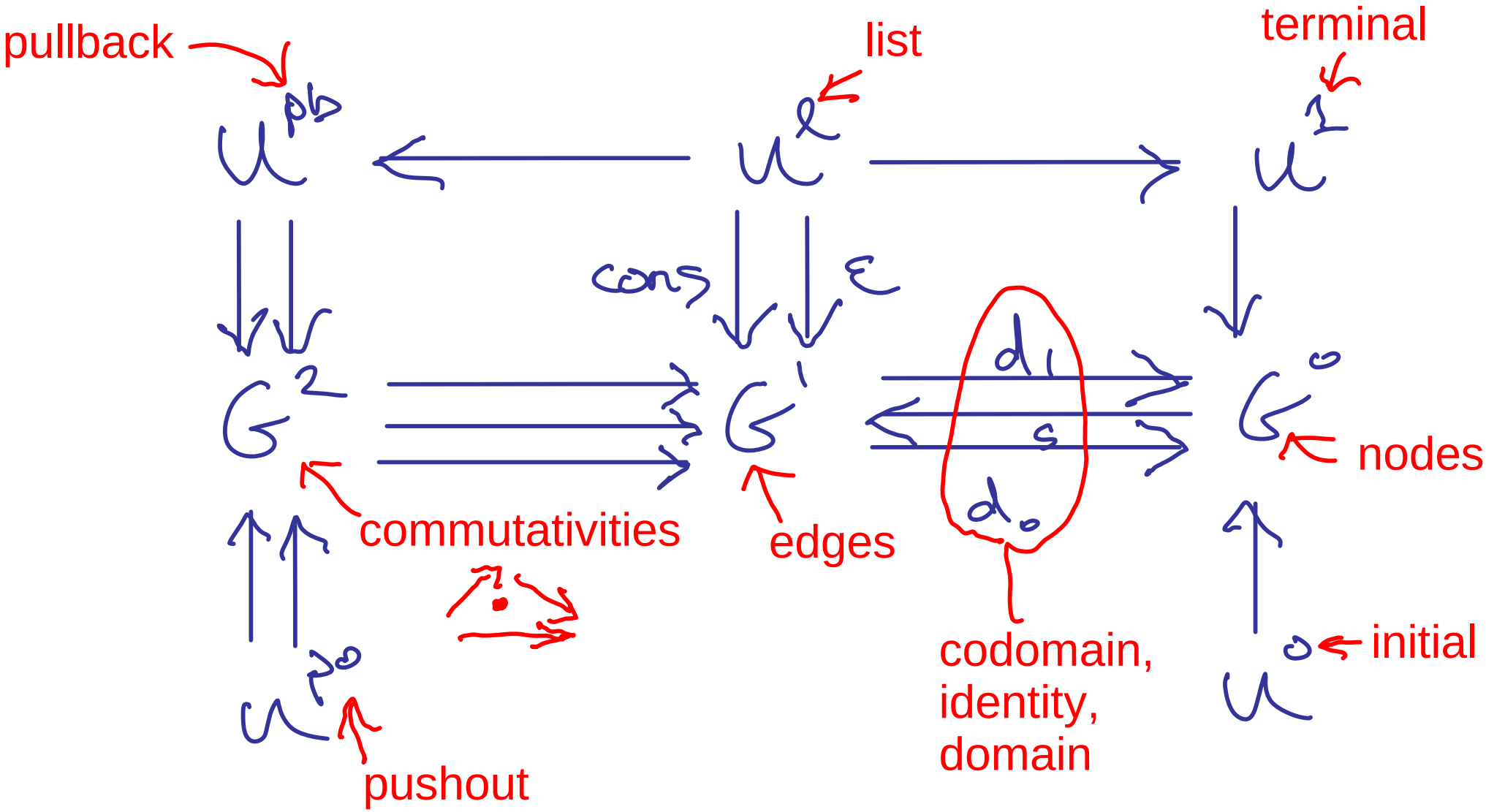
Strict model: the canonical product
- convenient for syntax (universal algebra)

Non-strict model: any product
- convenient for semantics

We find ways to reconcile these.

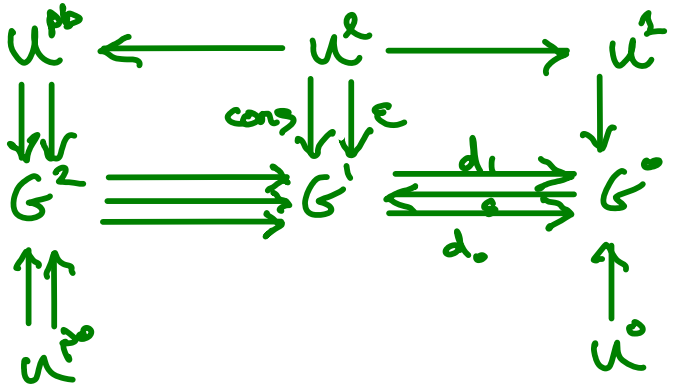
Sketches for AUs

G for graph, U for universals



There are equations amongst the operators here.

e.g. product universal, N is MxM



The two Ms are *the same node.*

T (terminal) is the image of an element of U^0.

Graph includes:

The two triangles are the images of two elements of G^2.

They are images of an element of U^pb, specifying the square is a pullback.

T terminal, square a pullback => (N, p1, p2) a product cone.

"Contexts" and strictness

For sketches in general:

Two universals of different kinds might specify same node.
e.g. X is both YxZ and List(A)
Possibly *many* non-strict models (YxZ and List(A) isomorphic),
no strict ones (YxZ = List(A)).

For better control of strictness:
ensure each node specified by at most one universal.
Then every non-strict model has a canonical strict isomorph.

e.g. contexts - sketches built up in finitely many extension steps
* adjoin fresh node
* adjoin fresh edge between old nodes
* adjoin fresh commutativity in old triangle
* adjoin universal specification for fresh node and edges (e.g. a pullback and its projections)

$\mathbb{T}_0 \subset \mathbb{T}_1$

From now on all sketches are contexts.

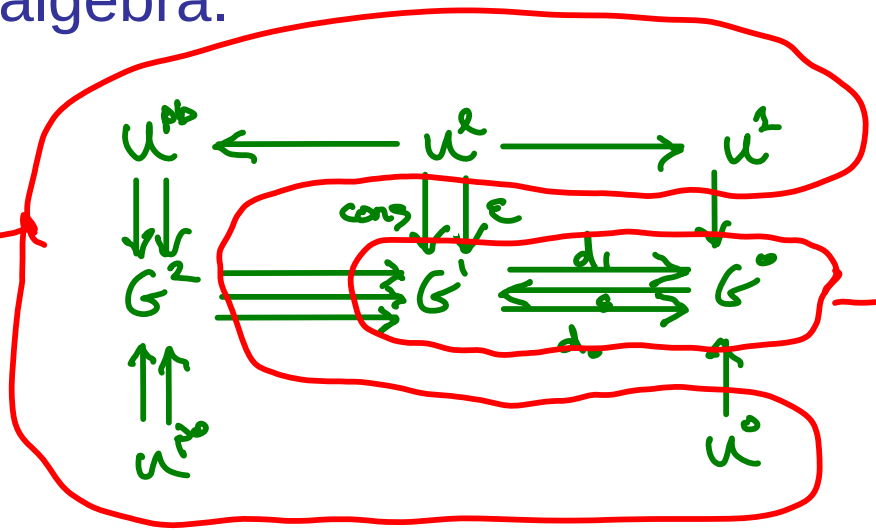
Sketch presents AU

$\mathbb{T} \longmapsto AU\langle\mathbb{T}\rangle$

classifying AU

(1) Sketch = generators + relations for AU.
Then use universal algebra.

relations



generators

(2) Equivalence extensions embody derivation rules for ingredients of sketches.
Make category of nodes and edges that can arise in equivalence extensions (modulo suitable equivalence).

Theorem It's isomorphic to AU<T>.

Sketch presents AU

(3) Sketch provides types, terms, equalities for a typed theory for the AU typed calculus.

TT derivation rules give types and terms for objects and morphisms of syntactic AU. (Maietti)

Conjecture It's isomorphic/equivalent to AU<T>.

Potential Simplify reasoning for sketches, using methods of dependent TT.

Difficulty Hard to match universal characterizations of two constructions.

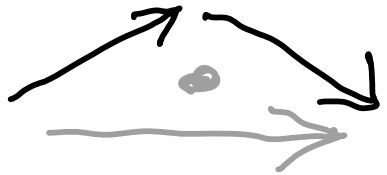
Equivalence extensions: derivation rules for sketches

$\mathbb{T}_0 \in \mathbb{T}_1$

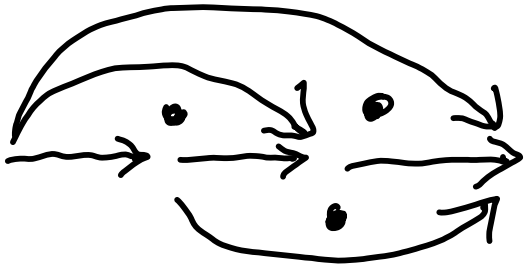
Given data in a sketch, $\mathbb{T}_0$

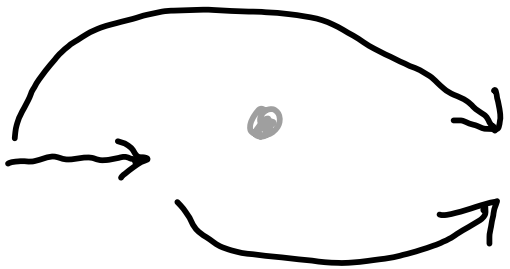
extend by adjoining delta: $\mathbb{T}_1$

composite edges

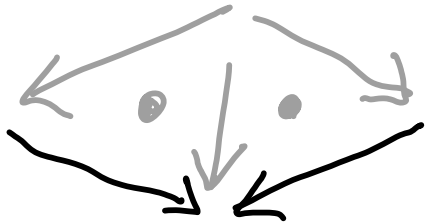


equality rules for edges, e.g. associativity





universal, e.g. pullbacks



+ pb universal

+ fillins for universals
+ uniqueness for fillins
+ inverses for certain edges, as required by categorical properties of AUs (balance, exactness, stability)

Object equalities

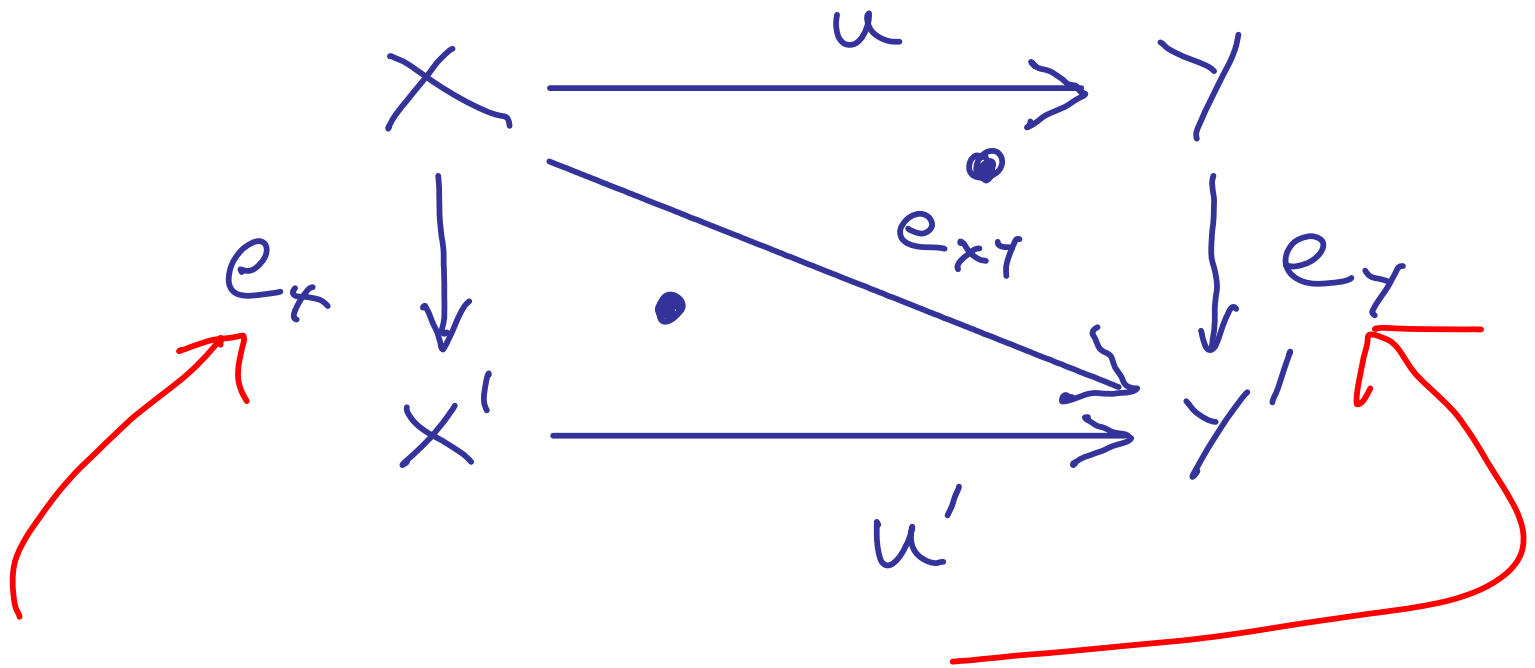
Equivalence extensions say *nothing* about equality of nodes.
We do that separately.

Inductively define certain edges to be object equalities.
* Identity edges are object equalities.
* Suppose have two universals of the same kind, with object equalities connecting the data. Then the fillin is also an object equality.

Object equalities are invertible and composable;
also (between any two nodes) unique.

Hence: equivalence relation on nodes.

Object equalities - between edges



two object equalities between nodes

$(e_x, e_{xy}, e_y, \bullet, \bullet)$ is object equality between edges u, u'

AU<T> is category of nodes, edges from equivalence extensions of T
- modulo object equalities.

Summary of first level of type theory: *within* AUs

Conjecture

* AU type theory (Maietti) can be related to sketches, giving alternative construction of AU<T> as syntactic category.

* That would allow calculus of dependent types to be applied in the equivalence extensions.

* Object equality an interesting example of type equality?

Second level of type theory: *amongst* AUs

Idea:

context = "context"

$\mathbb{T}_0$

contexts - sketches built up in finitely many extension steps
* adjoin fresh node
* adjoin fresh edge between old nodes
* adjoin fresh commutativitiy in old triangle
* adjoin universal specification for fresh node and edges (e.g. a pullback and its projections)

type in context = one of possible extension steps

extension by type = extension

$\mathbb{T}_0 \subset \mathbb{T}_1$

$\mathbb{T}_1$ $\quad$ $\mathbb{T}_0$

cf. categories with attributes

Get AU-functor

$AU\langle \mathbb{T}_0 \rangle \to AU\langle \mathbb{T}_1 \rangle$

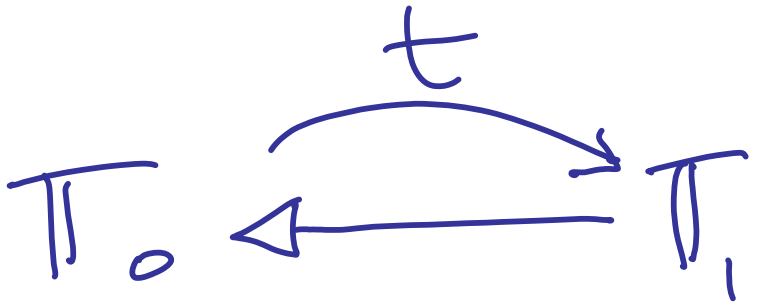
On models get transformation backwards, by model reduction.
Corresponds to context projection.

$Mod\text{-}\mathbb{T}_0 \leftarrow Mod\text{-}\mathbb{T}_1$

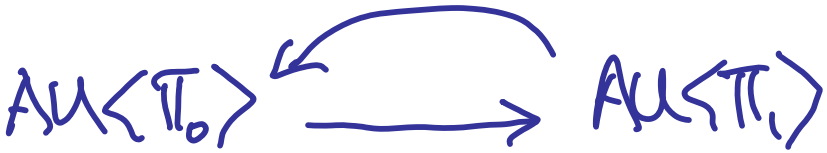
$\mathbb{T}_0 \leftarrow \mathbb{T}_1$

Terms?

Should be sections of context projections



We're looking for continuous t
- which should mean AU-functors backwards



= model

We construct these using the level-1 type theory (equivalence extensions of T_0)

The 2-category Con

Object = context

Morphism = context map = strict model in equivalence extension
(modulo object equality)
- in bijection with strict AU-functors between corresponding AUs

2-cells use homomorphisms of strict models

Has finite PIE-limits (products, inserters, equifiers), and pullbacks of extension maps (context projections).

Question Can we exploit the dependent type calculus in Con?
It would be a dependent type theory of generalized spaces
- not of sets
- nor of homotopy types

Type theory in Con?

Dependent TT is "point-set" in format:
"Here are the sets (types), here are their elements (terms)."

Sketch formalism is "point-free":
"Here are the ingredients you need to model in order to find a point."
- type (extension) indirectly describes space of models
- terms must reconstruct notion of point-free continuous map

Can we we fit TT formalism into Con?

Would restore points to point-free topology.
cf. use of geometric logic, despite incompleteness.
Slogan There are enough points if you look in the right place
(use generalized points, i.e. maps).

Goal: Dependent type theory of spaces

Concluding questions

Maietti already has a type theory for AUs.
Can this be securely related to the universal algebra via the sketches?
Is the TT syntactic category isomorphic to the algebraic classifying category?

How do the earlier Maietti-Vickers results relate to the new sketch theory?
Does the sketch-theoretic control of strictness help us understand the earlier problems with that?

Is there a genuine dependent type theory of spaces to be found round the 2-category of contexts?
How can we exploit it?
What is the scope of its mathematics?
(Should approximate geometric logic.)