Ecce ain't Epigram2

but I like it
What Ecce Is?

The Epigram Core Context Editor

A way for us to get our hands on Epigram's Innards...

But also a fully explicit dependently typed programming language in it's own right...

With some really neat features
Features of Ecce

- Bi-directional type Checking
- Aspect Oriented Type Theory
- Observational Equality
- Proof Irrelevant Universe of Propositions
- A closed Type Theory with a universe of data types
- Undo
Features of Ecce

- Bi-directional type Checking
- Aspect Oriented Type Theory
- Observational Equality
- Proof Irrelevant Universe of Propositions
- A closed Type Theory with a universe of data types
- Undo

There's a lot that it doesn't have too...

...Implicit syntax, much in the way of Elaboration
Bi-Directional Syntax

We separate those term formers whose types can be inferred, and those we must check.

ExTerms: Type constructors, references and variables, non-canonical forms, and eliminations.

InTerms: Canonical values
  - Constructors can inhabit more than one type
Aspect Oriented TT

We've split the implementation of the core not by function (parsing, type-checking, computation...) but by language feature.

The code for type checking a feature is in the same file as it's computation behavior and it's syntax.

Allows new features to be added, switched on/off, re-implemented very easily.

A source to source translator splices the functions back together for compilation.
The Epigram Core now implements Observational Equality

The equality operator unpacks under type constructors

So

\[(f : (x : X) \rightarrow T x) \iff (g : (x : X) \rightarrow T x)\]

Unpacks to:

\[(a, b : X) \rightarrow (a : X) = (b : X) \rightarrow (fa : Ta) \iff (gb : Tb)\]
Proof Irrelevance

Equality inhabits a proof irrelevant universe of propositions.

Closed under universal quantification (inhabited by lambda), conjunction.

We will also have functional quotient types
A Universe of Data

The core theory includes a universe of data declarations.

Actually 3 universes...

Enumerations - encode constructor choice

Signatures - encode tupling

Data - uses the first two to encode data
Closed Theory

This means that the core is a closed theory...

Contains all data-types you can define

Elim is a generic program, defined over the syntax of data definitions

Big step forward in defining Epigram in Epigram
The Proofstate

The Epigram Proof State is

Syntactically: hierarchical, definitions contain declarations and sub-definitions. Definitions are either meta-variables or type checked terms.

Semantically: flat, definitions are lambda-lifted over declarations.

Updates traverse syntax, which is local and so sharing is preserved.
Ecce Tactics

Ecce is a language of tactics for manipulation of the proof state and a term and proof state printer and parser.

- Introduce new declarations and definitions
- Upgrade, downgrade or delete definitions
- Move around the proof state

So far so basic, but the hope is to test high level elaboration by seeing it's effect on the proof state
Epigram 2

Editor

Elaborator

Core

Theorem Prover

Ecce