Let me formalize in Agda something I'd said earlier in email. Here is the original statement.

> I would view ornaments as a generalization of equivalences, and refinements as not really comparable to either. An equivalence is very precise and should be the easiest to automate. I think the Ko and Gibbons formulations of ornament (which I'm not sure is really "equivalent" to the McBride and Dagand formulation) is essentially an equivalence plus a refinement, so under this interpretation an equivalence could be seen as a degenerate ornament in which the refinement is trivial.

Let's define an equivalence; this is taken from *Equivalences for Free!* which in turn is taken from HoTT.

```
record IsEquiv {a b : Level} (A : Set a) (B : Set b) (f : A → B) : Set (a ⊔ b) where
  constructor isEquiv
  field
    f⁻¹ : B → A
    sect : (a : A) → (f⁻¹ ∘ f) a ≡ a
    retr : (b : B) → (f ∘ f⁻¹) b ≡ b
    adj : (a : A) → retr (f a) ≡ cong f (sect a)
open IsEquiv

record Equiv {a b : Level} (A : Set a) (B : Set b) : Set (a ⊔ b) where
  constructor equiv
  field
    func : A → B
    isEq : IsEquiv A B func
open Equiv
```

Here's the definition of heterogeneous relation from the Agda standard library. Note that it's the same as the one in Talia's picture of the blackboard, except it uses Set rather than Prop (although Agda now has its own Prop due to recent work by Jesper Cockx, based on the new sProp for Coq that Tabareau and other people are working on, done during Jesper's visit to Nantes).

```
REL : {a b : Level} → Set a → Set b → (ℓ : Level) → Set (a ⊔ b ⊔ lsuc ℓ)
REL A B ℓ = A → B → Set ℓ
```

When I think of a refinement however, I think of a type with a predicate, which is the following, and it's important that the codomain be Set in this case.

```
Pred : {a : Level} → Set a → (ℓ : Level) → Set (a ⊔ lsuc ℓ)
Pred A ℓ = A → Set ℓ
```

I would then define a refinement as a pair of a type and a predicate on that type. Let's create a new data type for clarity.

```
data Refinement {a ℓ : Level} (A : Set a) (P : Pred A ℓ) : Set (a ⊔ ℓ) where
  refinement : (x : A) → P x → Refinement A P
```

If Talia has a different formulation in terms of a relation, I'd be interested in seeing that.
We could describe ornamenting ℕ with a vector of booleans using a refinement as follows.

```
ornℕvecBool : Set
ornℕvecBool = Refinement ℕ (Vec Bool)
```

Now let me define an Ornament in a way that is similar to what Talia is talking about. It's too general, since it encompasses things that are not ornaments, and perhaps there are other flaws as well, but it's simple and will give us something to refer to.

Define an Ornament to be an equivalence between a refinement of A (the base type), where the refinement carries the extra information of the ornament, and B, which is the fully-ornamented type, as the following.

```
Ornament : {a b ℓ : Level} (A : Set a) → (B : Set b) → Set (a ⊔ b ⊔ lsuc ℓ)
Ornament {ℓ = ℓ} A B = Σ (Pred A ℓ) (λ P → Equiv (Refinement A P) B)
```

Now, returning to my opening quote, I think of ornaments as strict generalizations of equivalences. This first means the following type is inhabited.

```
Equivalence→Ornament : {a b : Level}{A : Set a} → {B : Set b} → Equiv A B → Ornament A B
Equivalence→Ornament eq = (λ _ → ⊤) , equivTrans RefA≃A eq
```

As can be seen the proof here is extremely simple, modulo some lemmas which are straigthfoward. Note the trivial refinement. The lemmas have been hidden from the generated PDF but are in the source file. I believe it would be straightforward as well to prove that every equivalance is an ornament in either the McBride/Dagand or Ko/Gibbons formulations.

By Propositions as Types (PAT), this is interpreted as: If there is an equivalance between A and B, then there is also an ornament between A and B.

On the other hand the converse is not true.

```
Ornament→Equivalence : {a b ℓ : Level}{A : Set a} → {B : Set b} → Ornament {ℓ = ℓ} A B → Equiv A B
Ornament→Equivalence orn = {! !}
```

There is no way to fill in the hole above. A counterexample is that there is an ornament from ℕ to *List Bool* but the two are not equivalent. Thus ornaments are a strict generalization of equivalences.

Similarly when I said refinements are not comparable to either, I meant there is no function from refinemennts to or from either equivalances or ornaments.

I believe what Talia was talking about is that equivalances can be defined in terms of relations, and ornaments can be defined in terms of equivalances (but as noted it will not be an equivalence of the original types).