binrel_com[12,41]

================
BINARY RELATIONS
================

There is a design choice here with choosing whether to build theories
of relations using first-order or higher-order syntax. For example, we
could write:

A)Refl(T;R) == $\forall x{:}T.\ R(x,x)$ (higher order)

or

B) Refl(T;x,y.R[x;y]) == $\forall x{:}T.\ R(x;x)$ (first order)

The approaches are equivalent.

The pros for A) are:
1. Consise definitions, no extra binding vars floating around.
   (Esp with things like symmetrize).
2. Treating relations as first class objects.

The pros for B) are:
1. Most binary relations are defined using 1st order syntax
   (subterm slots for args rather than args supplied using application)
   This gives much more flexibility with relation display forms.

2. Functionality lemmas can easily be proven that allow rewriting of
   R wrt <=>. With A) functionality lemmas are required for instances of
   of typed lambda terms and one would have to introduce special rw relations for
   equivalence of binary relations. (The rewrite package doesn't currently
   handle general higher order equivalence relations. (Relations on objects in
   function types where the relations on domain and range types are arbitrary
   equivalence relations.)

3. Abstraction expansion does appropriate substitutions when we have concrete
   instances of R.

For now, go with approach B), though approach A) is I think ultimately more
desirable.