Encoding Equivalences as Swaps

by Sven Nilsen, 2020

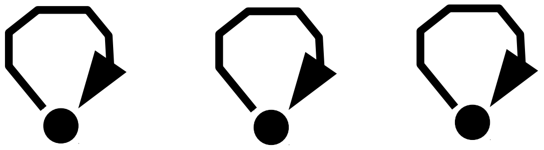
In this paper I show that equivalences can be encoded as swap operations.

Theorem proving where functions can be applied to equivalences has the following law:

f(a ~= b) == (f(a) ~= f(b))

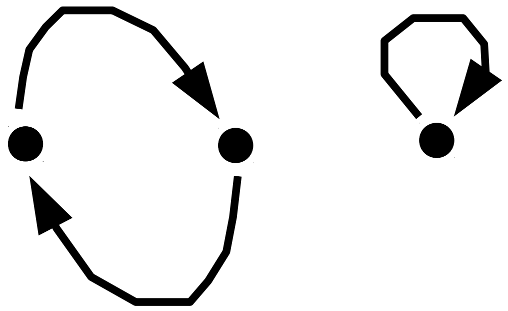
The semantics of such theorem proving is a generalization of "normal" theorem proving.

One key insight of this semantics is that in "normal" theorem proving,
every object is equipped with an arrow pointing to itself:



Since "normal" theorem proving only operates on objects, the arrow are unchanged.

Theorem proving with equivalences is a generalization where one is permitted to make arrows point from one object to another, but only according to certain rules.



For two objects to be equivalent, there must exist an isomorphism between the two. An isomorphism is a structure preserving map from one object to another, together with an inverse structure preserving map. By default, all objects with an arrow pointing to themselves are equivalent to themselves. When one creates an isomorphism between two objects, one can "forget" the original arrows, because an arrow pointing from an object back to itself can always be constructed by composing the new arrows. Likewise, if one creates more complex loops of arrows, one can "forget" the original loops. Therefore, the minimum number of arrows required to derive the remaining arrows is always a fixed number.

It turns out that when one swaps two elements, one is implicitly creating two such structure preserving maps to form an isomorphism! The semantics of an equivalence `a ~= b` is encoded into the representation of a swap operation.