CCSU
DEPARTMENT OF MATHEMATICAL SCIENCES

COLLOQUIUM

Friday, February 25
2:00 – 3:00 pm in MS 101

Type Isomorphism and Program Isomorphism

Carlos C. Martinez

Wesleyan University

Abstract

Unification has been a fruitful subject in logic and automated deduction. The unification problem can be stated as finding substitutions (instantiations of free variables) of two given terms that equalize the terms under some equality notion. Matching is the particular case when one of the terms is closed.

Two types A and B are said to be isomorphic if there is an invertible term having type A -> B. It is natural to want to be sensitive to type isomorphism when one is doing higher-order rewriting, in particular if one is interested in code transformation. For example, suppose one wants to perform a transformation with a certain function-pattern of type A -> (B -> C). The use of standard higher-order matching allows us to ignore the names of the arguments in a code fragment potentially matching the pattern. However, the order in which these parameters appear in the code is significant, since it determines the code's type. Since the type B -> (A -> C) is isomorphic to the original, a code fragment of this type may very well be a candidate that we want to consider.

Indeed, what we require is a richer notion of matching which accepts a match as long as the term being matched is the same as the target term modulo a type isomorphism. That is, type isomorphism induces a notion of equality on terms, more lenient than the usual equality, and it is this equality that we will use to guide our matching.

For further information:

gotchevi@ccsu.edu (860) 832-2839