Meeting: 1003, Atlanta, Georgia, AMS CP 1, AMS Contributed Paper Session

1003-68-1660 **Mendez Carlos Martinez*** (cmartinez@wesleyan.edu), 222 Church Street, Middletown, CT 06457. *Unification and Matching modulo Type Isomorphism.*

Unification algorithms have been a fruitful subject in automated deduction, the decision problem can be stated as the problem of finding instantiations of free variables (*substitutions*) of two given terms, that equalize the terms under some equality notion. On other hand Type isomorphism has been a subject with nice applications to software retrieval, we say that two types A, B to be isomorphic if there is an invertible term t having type $A \to B$. So if we are interested in code transformation, then the use of standard higher-order *matching* allows us to ignore the *names* of the arguments in a code fragment potentially matching the pattern, but the *order* in which these parameters appear in the code is significant, because it determines the code's type. Since the type $B \to (A \to C)$ is isomorphic to the original $A \to (B \to C)$. So it seems that a more refined tool than standard higher-order matching would be useful. 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*. To our knowledge this relation on terms has not been explored in the published literature. (Received October 06, 2004)