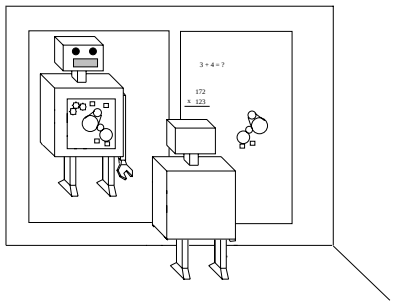
Theory of Program Self-Reflection

John Case (University of Delaware)
Award # CCR-0208616

The robot depicted below has a transparent front so that the entire *program* responsible for its behavior is visible in the mirror. It is copying its program, corrected for mirror reversal, on a board next to the mirror. Then it can employ, for its further calculations and decisions, a perfect program self-model. It has *usable self-knowledge*.



A robot acts in a world containing mirrors, but computer programs typically reside in the memory of modern digital computers where there are no mirrors. However, many programming languages permit a procedure (similar to self-replication in bacteria) with which programs can also obtain usable, perfect self-models.

The ancient Greek philosophers taught that self-knowledge is valuable. There are indications self-reflection is useful for robots. PI John Case sought *mathematical* insight into program self-reflection.

A property of programming languages *characterizes* program self-reflection if that property holds just in case program self-reflection is available. A property of many programming languages is availability of standard, easily understood *denotational control structures* (`dcs`es) such as looping, pipe-lining, and decision-branching. For example, pipe-lining provides for conveniently expressing the command to send the output of one program as input to another. A property of programming languages is *complementary* to program self-reflection if neither that property nor program self-reflection necessarily entails the other, but, if a programming language has both, it is most general as to availability of `dcs`es

The PI and his doctoral student Samuel E. Moelius III proved that, while availability of a class of `dcs`es can neither characterize [1] nor be complementary to program self-reflection [2], there is an easily understood, quasi-`dcs`, *coded pipe-lining*, which *is* complementary [2]. This control structure is like pipe-lining except that it *encodes* what is sent between programs.

Publications:

[1] J. Case and S. Moelius. Characterizing programming systems allowing program self-reference. *Computation and Logic in the Real World - Third Conference of Computability in Europe (CiE 2007), Proceedings*, volume 4497 of *Lecture Notes In Computer Science*. Springer, Berlin, to appear, 2007.

[2] J. Case and S. Moelius. Properties complementary to program self-reference. Submitted for publication, 2007.

Importance: These results about program self-reflection provide some very beginning insights regarding both what does not characterize its power and what is complementary to it.