**Infinite Computation, Co-induction and Computational Logic**

Co-induction provides the foundation for the lazy programming evaluation and type inference in functional programming. Co-induction is a technique for defining and proving properties of systems of concurrent interacting objects. It can be used as a proofing technique to show that an equation is satisfied by all possible implementations of such a specification. In general we use co-recursive functions along with lazy evaluation in Co-induction programming. Inductive definitions have 3 components: initiality, iteration and minimality.

Co-induction has been incorporated recently in logic programming in a semantic way. The traditional declarative semantics and operational semantics of logic programming is extended by co-inductive logic programming to allow reasoning over infinite and cyclic structures and properties. Co-inductive logic programming allows programmers to manipulate and work on infinite structures. As a result unification equations such as X = [ 1 | X ] is allowed in co-inductive logic programming.

Co-inductive Logical programming gives an operational semantics to declarative semantics that is based on greatest fixed point. Many believe that combination of inductive and co-inductive logic programming allows one to implement any desired LP semantics. But still they didn’t answer many problems yet.

Finally, this research paper on Co-induction by Dr. Richard Min and Dr. Gopal Gupta is a new era in co-induction Logical Programming.