UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE

http://smfi.unipr.it

SEMINARI di INFORMATICA

Relatore: Professor **Maximiliano Cristià**,
Universidad Nacional de Rosario &
CIFASIS, Rosario, Argentina

Luogo: Plesso di Matematica (aule A/C)

Tutti gli interessati sono invitati a partecipare
Prof. Gianfranco Rossi

Martedì 6 novembre[1], *9:00 - 10:00* Aula A, ***Introduction to Z specifications***

Martedì 6 novembre[2], *10:30 - 11:30* Aula A, ***Using Z specifications in practice***

Giovedì 8 novembre[3], *14:30 - 15:30* Aula C, ***Programs as formulas (not as proofs)***

1. *The Z specification language is one of the first formal languages intended for the specification of software systems. It is a language based on set theory and first-order logic. Z specifications take the form of state machines. In this seminar I will give a brief introduction to Z by means of practical examples. The goal of the seminar is to show to students that formal specification is helpful to understand what has to be implemented.*

2. *After the introduction to the Z formal notation given in the first seminar, in this second seminar I will show three verification activities that can be done having a Z specification. First, I will show how to animate Z specifications by means of the {log} tool, and why this is important. Then, I will show how to prove properties of Z specification, more precisely how to prove that a predicate is a state invariant. Finally, I will show that Z specifications can be used as the source for test cases with which later the implementation will be tested.*

3. *After the Curry-Howard correspondence and the type theories developed in the past decades, it emerged the idea of programs as proofs (of their properties). That is, if we have a proof P of a theorem T, we can make P a program whose type is T. This theory was a major breakthrough in programming language design and software verification. However, in this talk, I'll show an alternative and, in some sense, previous view: programs as formulas. In other words, let's make the very formula to be a program; that is, we don't need to make a proof to have a program. In particular I'm going to show {log} (setlog), which is a programming language but also an automated theorem prover. In {log} formulas are programs, and programs are formulas. We can use the same tool and the same language to specify programs, to simulate these specifications, to generate test cases and to prove properties of them. {log} works with formulas over the theory of finite sets, which allows for the specification of large classes of programs.*