Tutorial on λProlog

Amy Felty
INRIA

Elsa Gunter
Bell Laboratories

Dale Miller
University of Pennsylvania

Frank Pfenning
Carnegie Mellon University

The logic programming language λProlog extends Prolog by containing polymorphic types, higher-order programming, λ-terms as data structures, higher-order unification, modules, lexical scope, abstract data types, and implicational and universally quantified goals. These extensions are integrated into a single logical system and provide λProlog with expressive strengths not found in Prolog. After surveying various aspects of this language, we shall focus on its uses in specifying proof systems and implementing theorem provers. In particular, we shall discuss how the syntax and inference rules of object logics can be represented and on how tactic-style theorem provers can be written. We shall also illustrate how proofs can be built and manipulated in rather natural ways.

Tutorial on Equational Unification

Claude Kirchner

INRIA Lorraine & CRIN
Nancy, France

This tutorial presents unification, or equation solving in abstract algebras. After an introduction to the problems of the field, we present the general results on unification theory, including the decidability and unification type of various theories and classes of theories. Then, using a rule-based view of the equation-solving process, we show different aspects of the field. For syntactic unification we describe, in an abstract fashion, the classical algorithms of Robinson and Martelli-Montanari. Semantic unification (i.e., equational unification) is then studied. We first look at two examples, giving unification algorithms for commutative and associative-commutative theories. Then we focus on the combination of unification algorithms and on general equational unification. The notion of syntactic theory, for which unification algorithms can be automatically computed, is also introduced and its usefulness shown on several examples.