PRISM Revisited: Declarative Implementation of a Probabilistic Programming Language Using Delimited Control

Samer Abdallah

Jukedeck Ltd

PRISM is a probabilistic programming language based on Prolog augmented with primitives to represent probabilistic choice. PRISM is implemented using a combination of low level support from a modified version of B-Prolog, source level program transformation, and libraries for probabilistic inference and learning implemented in the imperative language C. More recently, developers of probabilistic languages working in the functional programming paradigm have taken the approach of *embedding* probabilistic primitives into an existing language, with little or no modification to the host language, primarily by using *continuations*: captured continuations represent pieces of the probabilistic program which can be manipulated to achieve a great variety of computational effects.

In this talk, I will describe an approach based on delimited control operators recently introduced into SWI Prolog. These are used to create a system of nested *effect handlers* which together implement a core functionality of PRISM—the building of explanation graphs—entirely in Prolog and using an order of magnitude less code. In addition, other declarative programming tools, such as constraint logic programming, are used to implement several tools for inference, such as the inside-outside and EM algorithms, lazy best-first explanation search, and MCMC samplers.

By embedding the functionality of PRISM into SWI Prolog, users gain access to its rich libraries and development environment. By expressing the functionality of PRISM in a relatively small amount of pure, high-level Prolog, this implementation will hopefully facilitate further experimentation with the mechanisms of probabilistic logic programming and extensions to new modelling features.