Embedding probabilistic languages

Chung-chieh Shan
Rutgers Univeristy

9/21/2010 at 10:30 am
CoRE A (Room 301)

Abstract

Probabilistic inference is popular in machine learning for writing programs that classify music and locate planes. It is also useful in cognitive science for modeling how people interpret words and recognize faces. However, it is applied much less than it could be, in part because most inference programs are written from scratch by hand. Instead, probabilistic models and inference procedures should be written as separate reusable modules. To this end, my coauthors and I have carried out and popularized a new approach that is easier to use and more expressive, namely to write models and inference in the same general-purpose language. Our approach makes deterministic parts of models run at full speed and lets inference procedures reason about themselves without interpretive overhead. The first half of this talk describes our approach and how we apply the same inference implementations to multiple realistic models, with performance competitive with the state of the art.

This approach to probabilistic programming exemplifies a recurring theme in my work: I apply programming-language theory to represent declarative knowledge as executable code instead of passive data. To further illustrate this theme, the second half of this talk reveals why our success in probabilistic inference was no fluke: anyone can invent such a representation, and many have, by applying a general pattern termed "finally tagless" to embed a domain-specific language into a general-purpose metalanguage. Applying this pattern in the face of uncertainty and adversity calls for two programming-language tools, whose theories I developed. First, because the control flow of the embedded language needs to shuffle, reinstate, and even duplicate in one fell swoop an arbitrary number of stack frames, the metalanguage should support first-class delimited continuations natively. Second, because the data flow of the embedded language needs to distinguish known values from yet-unknown ones, the metalanguage should manage variable bindings soundly. In domains ranging from probabilistic inference to natural-language semantics, these tools enhance modularity and turn manual reasoning into automatic

calculation.

Faculty Host: Michael L. Littman