Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 28 (2014) 80–99

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs

Action Editor: Paul Bello

A. Stuhlmüller [a,*], N.D. Goodman [b]

[a] Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, United States
[b] Department of Psychology, Stanford University, United States

Available online 25 July 2013

Abstract

A wide range of human reasoning patterns can be explained as conditioning in probabilistic models; however, conditioning has traditionally been viewed as an operation *applied to* such models, not *represented in* such models. We describe how probabilistic programs can explicitly represent conditioning as part of a model. This enables us to describe reasoning about others' reasoning using *nested* conditioning. Much of human reasoning is about the beliefs, desires, and intentions of other people; we use probabilistic programs to formalize these inferences in a way that captures the flexibility and inherent uncertainty of reasoning about other agents. We express examples from game theory, artificial intelligence, and linguistics as recursive probabilistic programs and illustrate how this representation language makes it easy to explore new directions in each of these fields. We discuss the algorithmic challenges posed by these kinds of models and describe how dynamic programming techniques can help address these challenges.

Keywords: Probabilistic programming; Social cognition; Recursive reasoning

1. Introduction

Reasoning about the beliefs, desires, and intentions of other agents—*theory of mind*—is a central part of human cognition and a critical challenge for human-like artificial intelligence. Reasoning about an opponent is critical in competitive situations, while reasoning about a compatriot is critical for cooperation, communication, and maintaining social connections. A variety of approaches have been suggested to explain humans' theory of mind. These include informal approaches from philosophy and psychology, and formal approaches from logic, game theory, artificial intelligence, and, more recently, Bayesian cognitive science. Many of the older approaches neglect a critical aspect of human reasoning—uncertainty—while recent probabilistic approaches tend to treat theory of mind as a special mechanism that cannot be described in a common representational framework with other aspects of mental representation. In this paper, we discuss how probabilistic programming, a recent merger of programming languages and Bayesian statistics, makes it possible to concisely represent complex multi-agent reasoning scenarios. This formalism, by representing reasoning itself as a program, exposes an essential contiguity with more basic mental representations.

Probability theory provides tools for modeling reasoning under uncertainty: distributions formalize agents' beliefs, conditional updating formalizes updating of beliefs based on evidence or assertions. This approach can capture a wide range of reasoning patterns, including induction and non-monotonic inference. In cognitive science, probabilistic methods have been very successful at capturing aspects of human learning and reasoning (Tenenbaum, Kemp, Griffiths, & Goodman, 2011). However, the fact that conditioning is an operation *applied to* such models and not itself *represented in* such models makes it difficult to

* Corresponding author.
E-mail addresses: ast@mit.edu (A. Stuhlmüller), ngoodman@stanford.edu (N.D. Goodman).

1389-0417/$ - see front matter © 2014 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.cogsys.2013.07.003

accommodate full theory of mind: We would like to view reasoning as probabilistic inference and reasoning about others' reasoning as inference about inference; however, if inference is not itself represented as a probabilistic model we cannot formulate inference about inference in probabilistic terms.

Probabilistic programming is a new, and highly expressive, approach to probabilistic modeling. A probabilistic program defines a stochastic generative process that can make use of arbitrary deterministic computation. In probabilistic programs, conditioning itself can be defined as an ordinary function within the modeling language. By expressing conditioning as a function in a probabilistic program, we represent knowledge about the reasoning processes of agents in the same terms as other knowledge. Because conditioning can be used in every way an ordinary function can, including composition with arbitrary other functions, we may easily express nested conditioning: we can condition any random variable, including random variables that are defined in terms of other conditioned random variables. Nested conditioning describes reasoning about reasoning and this makes theory of mind amenable to the kind of statistical analysis that has been applied to the study of mental representation more generally.

The probabilistic program view goes beyond other probabilistic views by extending compositionality from a restricted model specification language to a Turing-complete language, which allows arbitrary composition of reasoning processes. For example, the multi-agent influence diagrams proposed by Koller and Milch (2003) combine the expressive power of graphical models with the analytical tools of game theory, but their focus is not on representing knowledge that players' might have about other players' reasoning.

From a philosophical perspective, the nested conditioning approach supports the idea that the mental machinery underlying theory of mind is not necessarily a specialized module. Instead, it can be expressed in, and potentially acquired from, more general primitives for representing the world and means of composing these primitives. Indeed, the primitives needed to express conditioning as a probabilistic program are quite elementary: control structure, random choice, and recursive functions. This suggests that representations needed for complex theory of mind may be more elementary than often thought—though constructing and reasoning with these representations can be tricky.

The probabilistic programs we use here are essentially declarative probabilistic knowledge: they describe representations of (social) knowledge and inferences that follow, but they do not describe the process of inference itself. That is, as models of human cognition, the models we present here should be seen as a computational-level descriptions (Marr, 1982)—a normative abstraction that helps understand what kind of problem the human mind solves in social reasoning, not a proposal for a particular process or strategy used to solve these problems. Indeed, there are significant challenges to even computing the distributions implied by these models. We suggest algorithms for addressing the practical problems of modeling without intending to make a strong algorithmic claim about human cognition.

In the following, we first present background on computational modeling using probabilistic programs, then show examples of how programs with nested conditioning concisely express reasoning about agents in game theory, artificial intelligence, and linguistics. We then describe the challenges in computing the predictions of these models. Finally, we sketch a generic dynamic programming inference algorithm for probabilistic programs and explain how it can help address some of these practical challenges.

2. Representing distributions as probabilistic programs

A probabilistic program is a program in a universal programming language with primitives for sampling from probability distributions, such as Bernoulli, Gaussian, and Poisson. Execution of such a program leads to a series of computations and random choices. Probabilistic programs thus describe models of the stochastic generation of results, implying a distribution on return values. In our examples, we use Church (Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum, 2008), a probabilistic programming language based on the stochastic lambda calculus. This calculus is universal in the sense that it can be used to define any computable discrete probability distribution (Dal Lago & Zorzi, 2012) (and indeed, continuous distributions when encoded via rational approximation).

Church is a close relative of the functional programming language Scheme (Abelson & Sussman, 1996). In this language, function application is written in prefix notation:

```
(+ 3 2) → 5
```

The same applies to conditionals:

```
(if (> 3 2) true false) → true
```

Functions are first-class values and can be defined using λ. For example,

```
(λ (x) (* x 2)) → < function object >
```

refers to a function that doubles its argument. Values can be bound to variables using `define` and, for explicit scope, with `let`:

```
(let ([y 3]) (+ y 4)) → 7
```

For function definitions,

```
(define (double x) (* x 2))
```

is short for:

```
(define double (λ (x) (* x 2)))
```

The random primitive `(flip p)` samples from a Bernoulli distribution: it returns `true` with probability p, `false` with probability $1 - p$. By composing random primitives such as `flip` with deterministic computation, we can build complex distributions. For example, the expression