Science of Computer Programming 85 (2014) 115–136

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

An old new notation for elementary probability theory[☆]

Carroll Morgan[1]

University of New South Wales, NSW 2052, Australia

ARTICLE INFO

Article history:
Received 7 January 2013
Received in revised form 14 May 2013
Accepted 5 September 2013
Available online 7 October 2013

Keywords:
Probability theory
Eindhoven quantifier notation
Hidden Markov Models
Monads
Haskell

ABSTRACT

The Eindhoven approach to quantifier notation is 40 years old. We extend it by adding "distribution comprehensions" systematically to its repertoire; we believe the resulting notation for elementary probability theory is new.
After a step-by-step explanation of the proposed notational innovations, with small examples, we give as our exemplary case study the probabilistic reasoning associated with a quantitative noninterference semantics based on Hidden Markov Models of computation. Although that example was the motivation for this work, we believe the proposal here will be more generally applicable: and so we also revisit a number of popular puzzles, to illustrate the notation's wider utility.
Finally, we review the connection between comprehension notations and (category-theoretic) monads, and show how the Haskell approach to monad comprehensions applies to the distribution comprehensions we have introduced.

1. Context and motivation

Conventional notations for elementary probability theory, i.e. discrete distributions usually over finite sets, are more descriptive than calculational. They communicate ideas, but they are not algebraic (as a rule) in the sense of helping to proceed reliably from one idea to the next one: and truly effective notations are those that we can reason *with* rather than simply *about*. In our recent work on security, the conventional notations for probability became so burdensome that we felt that it was worth investigating alternative, more systematic notations for their own sake.

The Eindhoven notation was designed in the 1970s to control complexity in reasoning about programs and their associated logics: the forty years since then have shown how effective it is. But as far as we know it has not been used for probability. We have done so by working "backwards," from an application in computer security (Section 9.2), with the Eindhoven style as a target (Section 2). That is the opposite, incidentally, of reconstructing elementary probability "forwards" from first principles – also a worthwhile goal, but a different one.

We judge our proposal's success by whether it simplifies reasoning about intricate probabilistic structures in computer science and elsewhere. For that we give a small case study, based on noninterference-security semantics, both in the novel notation and in the conventional notation; and we compare them with each other (Section 9). We have also used the new notation more extensively [18].

Although the notation was developed retroactively, the account we give here is forwards, that is from the basics towards more advanced constructions. Along the way we use a number of popular puzzles as more general examples.

☆ This article extends an earlier publication [21].
E-mail address: carrollm@cse.unsw.edu.au.
1 I am grateful for the support of the Dutch NWO (Grant 040.11.303) and the Australian ARC (Grant DP1092464).

0167-6423/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.scico.2013.09.003

2. The Eindhoven quantifier notation, and our extension

In the 1970s, researchers at THE in Eindhoven led by E.W. Dijkstra proposed a uniform notation for quantifications in first-order logic, elementary set theory and related areas [5]. By $(\mathcal{Q}x : T \mid rng \cdot exp)$[2] they meant that *quantifier* $\mathcal{Q}$ *binds variable* x *of type* T *within textual scope* $(\cdots)$, *that* x *is constrained to satisfy formula* rng, *that expression* exp *is evaluated for each such* x *and that those values then are combined via an associative and commutative operator related to quantifier* $\mathcal{Q}$. These examples make the uniformity evident:

$$\begin{array}{rll}
(\forall x : T \mid rng \cdot exp) & \text{means} & \text{for all } x \text{ in } T \text{ satisfying } rng \text{ we have } exp,\\
(\exists x : T \mid rng \cdot exp) & \text{means} & \text{for some } x \text{ in } T \text{ satisfying } rng \text{ we have } exp,\\
(\Sigma x : T \mid rng \cdot exp) & \text{means} & \text{the sum of all } exp \text{ for } x \text{ in } T \text{ satisfying } rng,\\
\{x : T \mid rng \cdot exp\} & \text{means} & \text{the set of all } exp \text{ for } x \text{ in } T \text{ satisfying } rng.
\end{array}$$

A general shorthand applying to them all is that an omitted range $\mid rng$ defaults to $\mid true$, and an omitted $\cdot exp$ defaults to the bound variable $\cdot x$ itself.

These (once) novel notations are not very different from the conventional ones: they contain the same ingredients because they must. Mainly they are a reordering, an imposition of consistency, and finally a making explicit of what is often implicit: bound variables, and their scope. Instead of writing $\{n \in \mathbb{N} \mid n > 0\}$ for the positive natural numbers we write $\{n : \mathbb{N} \mid n > 0\}$, omitting the "$\cdot n$" via the shorthand above; the only difference is the explicit declaration of n via a colon (as in a programming language) rather than via $n \in \mathbb{N}$ which, properly speaking, is a formula (with both n and $\mathbb{N}$ free) and doesn't declare anything. And instead of $\{n^2 \mid n \in \mathbb{N}\}$ for the square numbers, we write $\{n : \mathbb{N} \cdot n^2\}$, keeping the declaration in first position (always) and avoiding ambiguous use of the vertical bar.

In program semantics one can find general structures such as

sets of distributions for probability and nondeterminism [25,17],
distributions of distributions for probabilistic noninterference security [18,19], and even
sets of distributions of distributions to combine the two [20].

All of these are impeded by the conventional use of "Pr" to refer to probability with respect to some unnamed distribution "of interest" at the time: we need to refer to the whole distribution itself.

And when we turn to particular instances, the semantics of individual programs, we need to build functions corresponding to specific program components. The conventional "random variables" are inconvenient for this, since we must invent a name for every single one: we would rather use the expressions and variables occurring in the programs themselves. In the small – but nontrivial – example of information flow (Section 9), borrowed from our probabilistic security work [18–20], we compare the novel notation (Section 9.2) to the conventional (Section 9.3) in those respects.

Our essential extension of the Eindhoven quantifiers was to postulate a "distribution comprehension" notation $\{\{s : \delta \mid rng \cdot exp\}\}$, intending it to mean "for all elements s in the distribution δ, conditioned by rng, make a new distribution based on evaluating exp." Thus we refer to a distribution itself (the whole comprehension), and we access random variables as expressions (the exp within). From there we worked backwards, towards primitives, to arrange that indeed the comprehension would have that meaning.

This report presents our results, but working forwards and giving simple examples as we go. Only at Definition 13 do we finally recover our conceptual starting point, a definition of the comprehension that agrees with the guesswork just above (Section 8.5).

3. Discrete distributions as enumerations

We begin with distributions written out explicitly: this is by analogy with the enumerations of sets which list their elements. The notation $f.x$ for application of function f to argument x is used from here on, except for type constructors where a distinct font allows us to reduce clutter by omitting the dot.

3.1. Finite discrete distributions as a type

A finite discrete distribution δ on a set S is a function assigning to each element s in S a (non-negative) probability $\delta.s$, where the sum of all such probabilities on S is one. The fair distribution on coin-flip outcomes $\{\mathsf{H}, \mathsf{T}\}$ takes both H, T to $1/2$; the distribution on die-roll outcomes $\{1..6\}$ for a fair die gives $1/6$ for each integer n with $1 \leqslant n \leqslant 6$. In general we have

[2] The original Eindhoven style uses colons as separators; the syntax here with $\mid$ and $\cdot$ is more in the Z style [11], and is one of many subsequent notational variations based on their innovation.