Monads for composing categorical and probabilistic meanings

Gianluca Giorgolo and **Ash Asudeh**
University of Oxford

In this paper we present a mathematical model for the interpretation of probabilistic meanings in a formal semantic framework. Our model is inspired by work in computer science for the definition of a semantics for probabilistic computations. The model is an instance of a more general approach we are exploring to give a constructive, and so computable, interpretation to a large number of semantic phenomena (Giorgolo and Asudeh 2012a; 2012b).

The question of combining categorical meaning representations with probabilistic ones is particularly important for natural language processing, where one often needs to combine information coming from diverse sources to assign an interpretation to a sentence. For instance we may want to combine the output of a stochastic classifier that assigns a certain probability to the assigned class with the output of a database lookup, which is usually a categorical response. Our model allows us to mix probabilistic meaning representations with purely symbolic ones and to propagate the probabilities from constituents to larger expressions in a well understood way.

In our model, meanings will be associated with a probability measure. For example an expression denoting an individual will refer to a specific individual with a certain probability and to others with different probability. In this way we can easily and compactly represent and keep track of ambiguous reference. Similarly if a predicate is applied to an argument the resulting proposition may be considered true with a certain probability p and false with probability $1 - p$.

The main advantages of our model are the following:

- It allows the combination of handcrafted symbolic lexical specifications with automatically generated meaning representation (e.g. statistically informed representation, or elaboration of sensors inputs),
- It does not force us to list all the combinatorial possibilities for the mixing of different sources of probabilistic meanings, providing us instead with a general mechanism to compute them on the fly,
- The approach can be generalized to accomodate other forms of enriched meanings, in particular other forms of uncertainty.

Our model for probabilistic meanings is based on category theory and more specifically on *monads*. Monads appear in category theory when studying the algebra of the endofunctors of a category. Besides their importance in category theory, monads have been proposed as a model for a number of notions of computations with side-effects, and, in the context of functional programming, as a way to define different notions of control. Our application of monads is very close to this use in computer science, as we will characterize probabilistic meanings as akin to non-deterministic computations, in contrast to traditional categorical meanings, which operate more similarly to standard (pure) functions. We assume that the reader is already familiar with monads. The interested reader may find a good categorial introduction in (Awodey 2010) and a more computationally oriented one in (Wadler 1992).

One of the main attractive features of a monadic model is that we can directly perform computations with it, freeing us from the need to define an additional representational level. We make monads part of our meaning language, together with the usual λ-calculus machinery that forms the internal language of our cartesian closed category.

In terms of composing meaning, we introduce the sequent calculus in figure 1, which is a linear logic version of the calculus presented in (Benton, Bierman, and de Paiva 1998). The formulae constrain how the linguistic elements can be combined together, and we assume that these constraints are determined by a lexicon and a syntactic analysis, as can be provided by a number of grammatical frameworks, such as Lexical Functional Grammar or some type of Categorial Grammar. We can prove a Cut elimination theorem for the calculus which gives us an effective way to compose meanings computationally. The formulae marked with the unary connective $\Diamond$ represent linguistic elements that have a monadic interpretation.

An interesting property of this compositional setup is its flexibility. In fact we can interpret the connective $\Diamond$ as representing different monads, or the composition of different monads.[1]

The model for our semantic framework is a cartesian closed category, whose objects are types and whose arrows are typed functions. The types we assume are all those built

[1] Monad composition is not a trivial matter. In general it is not possible to simply combine two monads to obtain a monad that represent the combination of the two, but there are different ways in which this can be done (Jones and Duponcheel 1993; Liang, Hudak, and Jones 1995).

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B} \multimap R \qquad \frac{\Delta \vdash t : A \qquad \Gamma, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u[y(t)/x] : C} \multimap L$$

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \eta(x) : \Diamond A} \Diamond R \qquad \frac{\Gamma, x : A \vdash t : \Diamond B}{\Gamma, y : \Diamond A \vdash y \star \lambda x.t : \Diamond B} \Diamond L$$

$$\frac{}{x : A \vdash x : A} id \qquad \frac{\Gamma \vdash B \qquad B, \Delta \vdash C}{\Gamma, \Delta \vdash C} Cut$$

Figure 1: Sequent calculus for a fragment of multiplicative linear logic enriched with a monadic modality, together with a Curry-Howard correspondence between formulae and meaning terms.

from a collection of atomic types, which includes at least a type for entities, a type for propositions, a type representing the measure space (in our case the interval $[0, 1]$) and a unit type, and closed under functional and product types.

The monad we use has been introduced by a number of researchers (Lawvere 1962; Giry 1982). Given that we are discussing natural processing applications we will restrict ourselves to the case of discrete probabilistic distributions. We will represent probability distributions using their graph format, i.e. set of pairs of elements and probability measures. The extension to continuous probability distribution requires us just to work directly with measure functions and substitute summations with integrals in our definitions.

The probability monad is defined by the triple $\langle P, \eta, \mu \rangle$, where P is the endofunctor of our category that maps each type X to the type $\mathcal{P}(X \times [0, 1])$, i.e. the set of subsets of the cartesian product of X with our measure space, and that maps each arrow $f : X \to Y$ to an arrow $P(f) : \mathcal{P}(X \times [0, 1]) \to \mathcal{P}(Y \times [0, 1])$ in the following way:

$$P(f)(p) = \{(f(x), a) \mid (x, a) \in p\} \tag{1}$$

The unit η and the join μ operations have very simple definitions. The unit maps a value to the trivial distribution that assigns to the event of observing that value the probability 1. The join can be understood in terms of conditional probability:

$$\eta(x) = \{(x, 1)\} \tag{2}$$

$$\mu(p) = \{(x, \sum_{(x,b) \in p'} ab) \mid (p', a) \in p\} \tag{3}$$

The same is true for the computationally friendlier counterpart of join, the bind $\star$ operation. We can define it in terms of P and μ:

$$p \star k = \mu(P(k)(m)) \tag{4}$$

or directly as the conditional probability of certain event y conditioned on the probability of an event x:

$$p \star k = \{(y, \sum_{(y,b) \in k(x)} ab) \mid (x, a) \in p\} \tag{5}$$

Linguistically this allows us to combine and propagate probabilities in a sensible way. In the case of purely symbolic meanings these will be lifted to the monadic level by the use of η, meaning that their contribution in terms of modifying probabilities will be empty (these values are certain). The truly probabilistic meanings will be combined by keeping track of their relative dependencies.

Tom	$\mathbf{t}$	e
John	$\mathbf{j}$	e
and	$\lambda p \lambda q.p \wedge q$	$t \to t \to t$
not	$\lambda p.\neg p$	$t \to t$
come	$\lambda x. \begin{cases} \{(T, 0.7), (F, 0.3)\} \text{ if } x = \mathbf{t} \\ \{(T, 0.2), (F, 0.8)\} \text{ if } x = \mathbf{j} \end{cases}$	$e \to P(t)$

Table 1: Toy lexicon

As an illustration, consider the toy lexicon in table 1. Here the only truly probabilistic meaning is represented by the entry for "come" which assigns different probabilities to event of coming depending on the entity involved. In the example, the entity "Tom" is quite likely to come while "John" is not. To compute the probabilistic truth value of a sentence like "Tom will come and John will not come" we first apply our calculus to obtain the derivation in figure 2. The resulting meaning evaluates to the following distribution (where x in (T, x) is the probability of truth and similarly for falsehood in (F, x)):

$$\begin{aligned} &[\![\text{come}]\!]\,([\![\text{john}]\!]) \star \lambda l.\,[\![\text{come}]\!]\,([\![\text{tom}]\!]) \star \\ &\lambda h.\eta([\![\text{and}]\!]\,(h)([\![\text{not}]\!]\,(l))) = \\ &\{(T, 0.56), (F, 0.44)\} \end{aligned}$$

which matches our intuition that the sentence represents a likely event.

However the likelihood assigned to the event denoted by the sentence seems quite low. This is because the probability mass is distributed between all possible ways in which the sentence can be made true or false, and there are three ways in which the sentence can be made false against a single possible way in which the sentence can be true. To correct this mismatch between probability and our intuitions, we can actually use a related monad, that instead of combining all events with the same outcome keeps all different "histories" separated. In terms of the model, we simply substitute sets of pairs of values and probability measures with lists of such pairs. Borrowing the syntax for lists and list comprehensions from the Haskell programming language, we can rewrite the

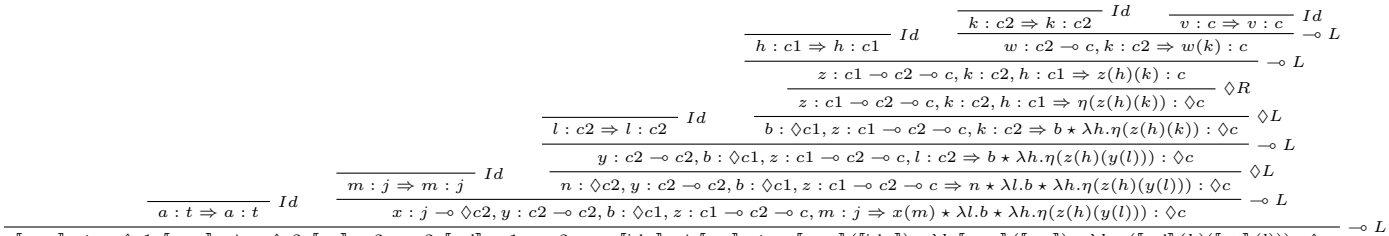

Figure 2: Proof for "Tom will come and John will not come"

defining elements of our monad as follows.

$$P(f)(p) = [(f(x), a) | (x, a) \in p] \quad (6)$$

$$\eta(x) = [(x, 1)] \quad (7)$$

$$\mu(p) = [(x, ab) \mid (p', a) \in p, (x, b) \in p'] \quad (8)$$

$$p \star k = [(y, ab) \mid (x, a) \in p, (y, b) \in k(x)] \quad (9)$$

There is also another way in which we can transform our model so that it tracks in a closer way the intuitions we have about uncertainty in natural language semantics. As noted by (Kuhnle 2013), in the case of a discrete probability distribution, we can interpret the associated probabilistic monad as the composition of the well known non-deterministic monad Set and the monad that associates each value with an element of a given monoid (often known as Writer). In the case of the monad formed by discrete probability distribution the monoid (which is actually a ring) is $(\mathbb{R}, \cdot, 1)$. Clearly we can substitute this monoid with another one, for instance one that mimics the (faulty) probability estimations that humans make. For example we could use the following (commutative) monoid whose elements represent discrete probabilities (Impossible, Low, Medium, High, Certain):

·	I	L	M	H	C
I	I	I	I	I	I
L		L	M	M	H
M			M	H	H
H				H	H
C					C

To sum up, we presented a mathematically well defined model for the composition of categorical meanings with meanings represented as probability distributions. Our model is based on a well understood and computationally well behaved mathematical construction, monads. The model has the advantage of allowing us to reuse handcrafted lexica with automatically generated ones. It is also flexible enough to accomodate other forms of uncertainty, as probability not always matches the intuitions of humans.

References

Awodey, S. 2010. *Category Theory*. Oxford University Press, second edition.

Benton, N.; Bierman, G. M.; and de Paiva, V. 1998. Computational type from a logical perspective. *Journal of Functional Programming* 8(2):177–193.

Giorgolo, G., and Asudeh, A. 2012a. $\langle M, \eta, \star \rangle$ Monads for conventional implicatures. In Aguilar Guevara, A.; Chernilovskaya, A.; and Nouwen, R., eds., *Proceedings of Sinn und Bedeutung 16*, volume 1, 265–278. MIT Working Papers in Linguistics.

Giorgolo, G., and Asudeh, A. 2012b. Missing resources in a resource-sensitive semantics. In Butt, M., and King, T. H., eds., *Proceedings of the LFG12 Conference*. Stanford, CA: CSLI Publications. 219–239.

Giry, M. 1982. A categorical approach to probability theory. In *Categorical aspects of topology and analysis, Proc. int. Conf., Ottawa 1981, Lect. Notes Math. 915, 68-85 (1982).*

Jones, M. P., and Duponcheel, L. 1993. Composing monads. Technical report.

Kuhnle, A. 2013. Modeling uncertain data using monads and an application to the sequence alignment problem. Master's thesis, Karlsruhe Institute of Technology.

Lawvere, F. W. 1962. The category of probabilistic mappings. Seminar handout.

Liang, S.; Hudak, P.; and Jones, M. 1995. Monad transformers and modular interpreters. In *In Proceedings of the 22nd ACM Symposium on Principles of Programming Languages. ACMPress*.

Wadler, P. 1992. Comprehending monads. In *Mathematical Structures in Computer Science*, 61–78.