Probabilistic dynamic semantics

Julian Grove and Aaron Steven White
University of Rochester

Abstract We introduce the framework of Probabilistic Dynamic Semantics (PDS), which we use to seamlessly integrate dynamic semantic analyses of lexical and discourse phenomena in the Montagovian tradition with probabilistic models of linguistic inference datasets. We show how PDS provides a general standpoint from which to understand uncertainty and context sensitivity in discourse and briefly illustrate applications to anaphora and vagueness.

1 Introduction

A core feature of dynamic semantic frameworks is their ability to precisely model how a linguistic expression's semantic value changes the context in which it is used (Karttunen 1976; Kamp 1981; Heim 1982; Heim 1983; Groenendijk and Stokhof 1990; Groenendijk and Stokhof 1991; Kamp and Reyle 1993; Asher 1993; Dekker 1993, i.a.). The integration of dynamic semantics with Montague semantics has in turn yielded a family of frameworks with which precise statements can be made about the way dynamic semantic values of complex expressions are compositionally derived from the morphemes that constitute them (Muskens 1996; de Groote 2006, i.a.). This integration has been further deepened with the introduction of powerful abstractions from programming language theory that allow a variety of semantic phenomena—indefiniteness, coreference, and quantification, among others—to be modeled in terms of distinct modules that themselves compose with each other in natural ways (Shan 2002; Barker 2002; Barker and Shan 2014; Charlow 2014; Bumford and Charlow 2022, i.a.).

One major strength of these frameworks, which we henceforth refer to under the heading of "Dynamic Montague Semantics" (DMS), is their ability to model a rich array of interpretive ambiguities. For example, from (1), one is quite likely to infer that the magician's assistant is secretly pleased, but not necessarily that the magician is pleased, even though, in principle, it may be that both are, or even that only the magician is.

(1) a. Whenever anyone laughed, the magician scowled and their assistant smirked.
    b. They were secretly pleased.

These inferences can be explained by invoking an ambiguity in what *they* can be used to refer to when it is preceded by (1a). In DMS, this ambiguity is commonly explained by positing that pronouns *read* representations of referents from representations of contexts and that ambiguities ensue due to the availability of multiple readable referent representations (Karttunen 1976, et seq.). In the case of (1), a DMS analysis might posit (i) that referent representations for a laugher, the magician, the assistant, and all subgroups of the three are added to some representation of

(1a)'s context of utterance; (ii) that *they* is ambiguous between being a singular and a plural pronoun (or is simply underspecified for number); and (iii) that after the contextual representation is changed by (1a), (a) only the individual referent representations of the magician and their assistant are readable if *they* is interpreted as having an individual referent; and (b) only the referent representation of the group containing both the magician and their assistant is readable if *they* is interpreted as having a group referent. Ultimately, there are three possible interpretations, none of which involve a laugher necessarily being pleased.

Two properties of this explanation are important. First, it understands the interpretation process as compositional. Without compositionality, it is difficult to explain why there is an apparent referential ambiguity in the first place. Second, uncertainty about the inferences to be drawn is explained as a consequence of *summing over* interpretations, which may jointly support only a subset of the inferences licensed by each interpretation on its own. The fact that one does not necessarily infer that the magician—or for that matter, their assistant—was pleased results from three distinct interpretations being summed over: one implying that the magician was pleased, one implying that their assistant was pleased, and a third implying that both were.

While DMS provides powerful tools for specifying the range of semantic values a linguistic expression *can* have—and thus the inferences it must and may support—it does not typically provide a way of specifying the likelihood that an expression *will* have a particular semantic value in a particular context. For example, it does not model the likelihood that *they* will refer to an individual versus a group in (1b). As a consequence, DMS does not make predictions about the likelihood that a language comprehender will draw a particular inference.

A natural place to look when approaching this challenge is the probabilistic semantics and pragmatics literature (see Erk 2022, for a recent review). In this literature, semantic values can take a variety of forms, such as probabilistic programs (Goodman and Lassiter 2015) or graphical models (Erk and Herbelot 2023), and can naturally state the likelihood of an inference in terms of the likelihood of particular interpretations of an expression. The main downside of these approaches is that they do not (currently) allow one to take up the modular approach to theory development that makes DMS so powerful. For instance, if one wants to combine a particular analysis of indefiniteness, coreference, and/or quantification with a probabilistic semantics, one must integrate them "by hand"; it is not simply a matter of defining a module that can be plugged into an existing theory or framework.

The aim of this paper is to introduce a framework—*Probabilistic Dynamic Semantics* (PDS)—that supports this sort of plugability as a means to seamlessly synthesize insights from both the DMS literature and the probabilistic semantics and pragmatics literature. The main idea behind the framework is to view probabilistic reasoning as arising from a module that can be added to a semantic theory without fundamentally changing its structure. PDS has three important features:

- *Compositionality*: analyses of complex expressions incorporating probabilistic uncertainty are derived from analyses of simpler expressions incorporating probabilistic uncertainty, using the type of composition scheme prevalent in DMS.
- *Modularity*: traditional semantic analyses may be *lifted* into the probabilistic setting without tampering with their core structure.
- *Formal evaluability*: in addition to being applicable to the traditional sources of data that inform the development of semantic theory (i.e., informal inference and acceptability judg-

ments), analyses couched within PDS can be quantitatively compared with one another by evaluating their relative fits to experimental data using standard model comparison metrics.

The first and second features are underwritten by the fact that the probabilistic reasoning components of PDS are developed using the same sorts of powerful abstractions from programming language theory that modern DMS approaches use to ensure modularity. Because of this, we can ensure that these probabilistic reasoning components may be well integrated into existing DMS frameworks. At the same time, these two features characterize some existing probabilistic semantic frameworks to varying extents. Bernardy, Blanck, Chatzikyriakidis, and Maskharashvili (2022), for example, introduce a dependently typed language which can be used to carry out probabilistic reasoning, and Grove and Bernardy (2023) present similar ideas in a modular format using λ-calculus and continuations.

The third feature—formal evaluability—is unique among DMS frameworks. PDS supports this feature by regarding every semantic representation as leading a double life. Semantic representations always have a discrete structure familiar from DMS. This discrete structure determines the range of interpretations—and thereby, the range of inferences—supported by a particular discourse. But PDS representations can always be *evaluated* to a probabilistic model in a way that retains important aspects of the discrete structure while also supporting fine-grained weightings of the inferences which that structure determines.

The methodological benefit afforded by this duality is that one may develop and test semantic analyses using quite fine-grained aspects of the distributions of judgments in formally collected experiments, including, e.g., variability in how comprehenders approach a particular task. This capability is itself supported by PDS's modularity; the way that PDS evaluates discrete structures into probabilistic models allows these models to be composed with further processes linking distributions on inferences with distributions on judgments about those inferences. PDS makes these *linking models* explicit by encoding them as functions from answers to a question under discussion (QUD; Ginzburg 1996; Roberts 2012) onto possible responses associated with some data collection instrument (e.g., a Likert scale, slider, etc.). Because they are defined explicitly (in type theory, no less), linking models can be stated and studied in their own right, independent of the particular semantic analysis under investigation.

In the next section, we give an overview of our proposals before formalizing them in Section 3. We illustrate a number of example analyses using the framework in Section 4 before concluding in Section 5 with a discussion of some prospects for applying and developing it.

Before we dive in, we should stress that our aim in this paper is to lay out the formal details of the framework of PDS, as well as to suggest possible applications of it that might be evaluated in future formal experiments. We take this tack for expository reasons: introducing the framework's mathematical underpinnings, analyses developed within it, and specific formal experiments to which we fit those analyses would prove unwieldy.[1] Instead, we pull back a little, in order to characterize the more general research program that we believe PDS represents.

[1]For an example application of the framework, see Grove and White 2024.

2 High-level sketch

DMS often models utterance meanings as maps from discourse states into sets of discourse states.[2] PDS inherits this functional view of utterances; but following much work in the probabilistic semantics and pragmatics literature (van Benthem, Gerbrandy, and Kooi 2009; Lassiter 2011; Frank and Goodman 2012; Zeevat 2013; Lassiter and Goodman 2017; Bergen, Levy, and Goodman 2016, i.a.), it translates this idea into a probabilistic setting: in PDS, utterances denote maps from discourse states to *probability distributions* over discourse states. Thus in comparison to DMS, PDS introduces a weighting on discourse states, allowing one to model preferences for certain resolutions of ambiguity over others.

2.1 Probability distributions as monadic values

In and of itself, this extension is not novel. More novel is that we view probability distributions as *monadic values* that inhabit types arising from a *probability monad* (see Giorgolo and Asudeh 2014; Bernardy, Blanck, Chatzikyriakidis, Lappin, et al. 2019; Grove and Bernardy 2023). We formalize this view in Section 3.2; but the gist is that viewing probability distributions this way allows PDS (i) to map linguistic expressions of a particular type to probability distributions over objects of that type so that the usual compositional structure of semantic analyses is retained; and thereby (ii) to compose probabilistic analyses with other DMS analyses of, say, anaphora; as well as (iii) to define explicit *linking models* that map probability distributions over discourse states to probability distributions over judgments recorded using some response instrument.[3]

Crucial for PDS is that because probability distributions are characterized by a monad, they may themselves be *stacked* while retaining properties important for semantic composition.[4] That is, the types derived from a probability monad may be inhabited by distributions over familiar types of objects—entities, truth values, functions from entities to truth values, and the like—or they may be inhabited by distributions *over* such distributions. And this stacking can be as deep as is necessary to model the sorts of uncertainty of interest to the analyst.

2.2 Two kinds of uncertainty

In the current paper, we argue that at least two levels of stacking are necessary in order to appropriately model two kinds of interpretive uncertainty, respectively, which we refer to as *metalinguistic* (or *type-level*) *uncertainty* and *occasional* (or *token-level*) *uncertainty*. Metalinguistic uncertainty is any kind of uncertainty which relates to lexical, structural, or semantic (e.g., scopal) ambiguity. For example, a polysemous word gives rise to metalinguistic uncertainty. Based on the content of its direct object, *ran* in (2) seems likely to take on its locomotion sense, though it remains plausible that it has a management sense if Jo is understood to be the race's organizer.

[2]In its distributive implementations, that is (for a discussion of distributive vs. non-distributive variants of dynamic semantics, see, e.g., Charlow 2019).

[3]This type of capability is often discussed in the experimental linguistics literature under the heading of *linking hypotheses* or *linking assumptions* (see Phillips et al. 2021). For our purposes, we define linking models to be statistical models that relate a PDS analysis (which determines a probability distribution over the inferences supported by a linguistic expression) to comprehenders' judgments, as recorded using a particular instrument.

[4]More to the point, monads give rise to *functors*, which are composable, giving rise to the "stacking".

(2) Jo ran a race.

In contrast, occasional uncertainty is that which is associated with an expression in view of some *fixed* meaning it has. Vague adjectives give rise to occasional uncertainty, for example, as witnessed by the vague inferences they support: the minimum degree of height *tall* requires to hold of entities of which it is true remains uncertain on any use of (3), even while the adjective's meaning plausibly does not always vary across such uses.

(3) Jo is tall.

In general, we conceptualize occasional uncertainty as reflecting the uncertainty that one has about a given inference at a particular point in some discourse, having fixed the meanings of the linguistic expressions.

Put slightly differently, metalinguistic uncertainty is a property of one's knowledge about the meanings of expressions *qua* expressions. Sometimes *run* means this; sometimes it means that. Thus, any analysis of the uncertainty about the meaning of *run* should capture that it is uncertainty about *types* of utterance act. In contrast, occasional uncertainty encompasses any semantic uncertainty which remains, having fixed the type of utterance act—it is uncertainty pertaining to the semantically licensed inferences themselves.[5]

To capture this idea, our approach regards these types of uncertainty as interacting with each other in a restricted fashion by taking advantage of the fact that distributions may be stacked. Because metalinguistic uncertainty must be resolved in order for one to draw semantically licensed inferences from uses of particular expressions, we take metalinguistic parameters to be *fixed* in the computation of occasional uncertainty. As we describe in Section 4, this rigid connection among sources of uncertainty is a natural consequence of structuring probabilistic reasoning in terms of stacked probability distributions. We show how PDS can capture the distinction between levels of uncertainty in Section 4.

2.3 Discourse states

We follow a common DMS practice by regarding discourse states as lists of parameters. We depart slightly from the usual assumption that these lists are homogenous by treating them as potentially arbitrarily complex, i.e., *polymorphic* (though, see Grove 2019; Bumford and Charlow 2022). As such, they could be structured according to a variety of models sometimes employed in formal pragmatics (e.g., Farkas and Bruce 2010). For example, we will define one parameter of this list to be a representation of the Stalnakerian common ground (or more aptly, the "context set": Stalnaker 1978, et seq.) and another parameter to be the QUD stack (Roberts 2012, et seq.).

We represent common grounds as probability distributions over indices encoding information about possible worlds, as well as what we call *contexts*. The possible world part of an index represents facts about how the (non-linguistic) world is—e.g., a particular individual's height—while the context part encodes certain facts about lexical meaning—e.g., the identity of the height threshold conveyed by a vague adjective such as *tall* (see Kennedy and McNally 2005; Kennedy 2007; Lassiter 2011, i.a.).

[5]See Beaver 1999; Beaver 2001, which describe an analogous bifurcation of orders of pragmatic reasoning in the representation of the common ground.

Utterances—and more broadly, discourses—map tuples of parameters onto probability distributions over new tuples of parameters. Moreover, complex linguistic acts may be *sequenced*; in general, the effect on an ongoing discourse of multiple linguistic acts may be computed by using the sequencing operation (*bind*) native to the probability monad. In this sense, compositionality of interpretation obtains in PDS from the level of individual morphemes all the way up to the level of complex exchanges. For example, a discourse may consist in (i) making an assertion, which (perhaps, under a simplified model) modifies the common ground; (ii) asking a question, which adds a QUD to the top of the QUD stack; or (iii) a sequence of these. Regardless, we require the functions encoding discourses to return probabilistic values, in order to capture their inherent uncertainty.

2.4 Linking models

A linking model takes a discourse as conceived above, together with an initial probability distribution over discourse states, and links them to a distribution over responses to the current QUD. The possible responses to the QUD are determined by a data collection instrument, which could be a Likert scale, a slider scale, or something else. Furthermore, the *distribution* over responses is fixed by a likelihood function whose choice is constrained by the nature of the values encoded by the instrument. Thus a Bernoulli distribution for instruments that produce binary values; a categorical distribution for instruments that produce unordered, multivalued discrete responses; a linked logit distribution for instruments that produce ordered, multivalued discrete responses; and so on.

3 Formalizing the framework

We now fill in the formal details of the above sketch. First, in Section 3.1, we introduce the basic aspects of our grammar formalism, which is combinatory categorial grammar (CCG; Steedman 1996; Steedman 2000). Then, in Section 3.2, we equip grammatical operations with probabilistic effects and show how probabilistic uncertainty in discourse can be modeled as a kind of probabilistic update on discourse states.

3.1 Monadic CCG

CCG is a highly lexicalized grammar formalism, in which expressions are equipped with syntactic types—i.e., *categories*. Syntactic types in CCG encode an expression's selectional and distributional properties. A noun phrase such as *a race*, for example, may be given the type *np*, while a determiner—something which, in English, occurs to the left of a noun in order to form a noun phrase—may be given the type np/n. Thus the forward direction of the slash indicates that a noun should occur to the *right* of the determiner.

We use CCG in our presentation of PDS because one of our goals is to write *semantic grammar fragments* which produce analyses of a given collection of probabilistic semantic phenomena. Having a grammar fragment (e.g., one which generates the stimuli about which inference judgments are experimentally collected to create some linguistic dataset) allows one to implement an unbroken chain that connects the semantic analysis of some phenomenon to a probabilistic model of judgments about expressions featuring the phenomenon. CCG is likely to be sufficiently expressive to capture most (if not all) of the kinds of syntactic dependencies found in natural languages (Joshi 1985; Vijay-Shanker and Weir 1994, et seq.; cf. Kobele 2006). Meanwhile, because it is semantically transparent, it makes writing such grammar fragments relatively straightforward.

In fact, the formalism we introduce is a proper generalization of CCG: monadic CCG. Monadic CCG equips the syntactic type system of CCG with a semantics that benefits from the expressive power of *monads*. Monads have provided a useful abstraction in linguistic semantics since the work of Shan (2002), who shows how they can be used to associate the semantic values of expressions with *effects*; that is, aspects of meaning—like intensionality, non-determinism, and quantification—that often complicate semantic composition as it is carried out in static systems based on functional application (see, e.g., Heim and Kratzer 1998). Much work since Shan's has used monads to implement different dynamic effects, incorporating notions like environment and state to model conventional implicature, anaphora resolution, and intensionality, as well as notions like non-determinism and partiality to model the semantic contribution of indefinites and presupposition triggers (see Unger 2012; Charlow 2014; Charlow 2020b; Charlow 2020a; Elliott 2022; Grove 2022). Our current goal is to show that arbitrary monadic semantic regimes can be seamlessly combined with CCG without affecting its syntactic type system (specifically, in virtue of a monadic type homomorphism). As a result, we can write semantic analyses which enjoy the formal advantages of CCG, along with the expressive power of monads.

3.1.1 CCG

For the purposes of this paper, we assume the particular set of atomic syntactic types in Definition 3.1.

Definition 3.1 (Atomic categories)**.**

$$\mathcal{A} ::= np \mid n \mid ap_{mod.} \mid d \mid s \mid sc \mid q_{ind.} \mid q_{deg.}$$

Thus while we have the usual categories for noun phrases (np), nouns (n), and sentences (s), we also add a few more: one for individual questions ($q_{ind.}$), one for degree questions ($q_{deg.}$), one for small clauses (sc), one for adjective phrases headed by modal adjectives such as *likely* ($ap_{mod.}$), and one for degree-denoting expressions themselves (d). The categories for questions will become important in Section 3.2 (specifically, Section 3.2.6 and later), while the categories for adjective phrases, degree-denoting expressions, and small clauses will feature in the analyses provided in Section 4. In general, given a set of atomic syntactic types $\mathcal{A}$, the set $C_{\mathcal{A}}$ of syntactic types over $\mathcal{A}$ is the smallest superset of $\mathcal{A}$ such that $b/a, b\backslash a \in C_{\mathcal{A}}$ just in case $a, b \in C_{\mathcal{A}}$.

Definition 3.2 (Categories over $\mathcal{A}$)**.**

$$C_{\mathcal{A}} ::= \mathcal{A} \mid C_{\mathcal{A}}/C_{\mathcal{A}} \mid C_{\mathcal{A}}\backslash C_{\mathcal{A}}$$

Thus following Definition 3.1, $C_{\mathcal{A}}$ includes the five elements of $\mathcal{A}$, as well as

$$s/np, s\backslash d, np/q_{ind.}, (sc\backslash d)/np, ap_{mod.}\backslash(q_{deg.}/np),$$

and so on. Any complex syntactic type in $C_{\mathcal{A}}$ features slashes, which indicate on which side an expression of that type takes its argument. Thus an expression of type b/a (for some two types

$$\dfrac{}{\Gamma, x : \alpha \vdash x : \alpha}\,\mathsf{Ax} \qquad \dfrac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \to \beta}\,{\to}\mathsf{I} \qquad \dfrac{\Gamma \vdash t : \alpha \to \beta \qquad \Gamma \vdash u : \alpha}{\Gamma \vdash t(u) : \beta}\,{\to}\mathsf{E}$$

$$\dfrac{}{\Gamma \vdash \diamond : \diamond}\,\diamond\mathsf{I} \qquad \dfrac{\Gamma \vdash t : \alpha \qquad \Gamma \vdash u : \beta}{\Gamma \vdash \langle t, u \rangle : \alpha \times \beta}\,{\times}\mathsf{I} \qquad \dfrac{\Gamma \vdash t : \alpha_1 \times \alpha_2}{\Gamma \vdash \pi_j(t) : \alpha_j}\,{\times}\mathsf{E}_j$$

Figure 1: λ-terms.

a and b) occurs with an expression of type a on its right in order to form an expression of type b, while an expression of type $b\backslash a$ occurs with an expression of type a on its *left* in order to form an expression of type b. We adopt the convention of notating syntactic types without parentheses when possible, under the assumption that they are left-associative; i.e., $a \mid_1 b \mid_2 c \overset{\text{def}}{=} (a \mid_1 b) \mid_2 a$ (where $\mid_1$ and $\mid_2$ are either forward or backward slashes). Thus for example, the type $s\backslash(np/np)$ continues to be written as such, while the type $(s\backslash np)/np$ may be shortened to '$s\backslash np/np$'.

To write CCG expressions, we use the notation

$$\begin{array}{c} s \\ \boxed{m} \end{array} \vdash c$$

which is to be read as stating that string s has category c and semantic value m. We assume s to be a string over some alphabet Σ (i.e., $s \in \Sigma^*$), which we regard as a finite set; e.g., the set of "morphemes of English". Meanwhile, we assume m to be a typed λ-term. We leave somewhat open the question of what types of λ-terms may be used to define semantic values, but we adopt at least the typing rules in Figure 1. Assuming that all semantic values are closed terms, we therefore have abstractions ($\lambda x.t$), applications ($t(u)$), and n-ary tuples ($\langle t_1, \cdots, t_n \rangle$), along with the empty tuple $\diamond$. We additionally assume that λ-terms can feature constants, drawn from some countable set.

As for the semantic types themselves, we can assume that there are the atomic types of Definition 3.3, where e is the type of entities; t is the type of truth values T and F; and ι is the type of intensional indices, whose identity we leave abstract for now (see Section 3.2, where we treat this type in depth).

Definition 3.3 (Atomic types).

$$A ::= e \mid t \mid \iota \mid \gamma$$

Furthermore, γ is the type of *lists* of entities of type e, which will be useful when we model discourse referents (DRs) as elements of such lists (see, e.g., Heim 1982; Dekker 1994; Unger 2012; Charlow 2014).

The full set of types relevant to Figure 1 is then formed from these atomic types, according to Definition 3.4.

Definition 3.4 (Types over A).

$$\mathcal{T}_A ::= A \mid \mathcal{T}_A \to \mathcal{T}_A \mid \mathcal{T}_A \times \mathcal{T}_A \mid \diamond$$

Just as with complex syntactic types, we adopt the convention of notating complex semantic types without parentheses when possible. Unlike syntactic types, we assume semantic types are right-associative.[6]

In CCG, expressions are combined to form new expressions using application rules, as well as composition (**B**) rules and (often) type-raising (**T**) and substitution (**S**) rules. The string *every linguist* can be derived by *right* application from expressions for the strings *every* and *linguist*, for example.

(4)
$$\dfrac{\overset{\textit{every}}{\boxed{\lambda p, q, i.\forall y.p(y)(i) \to q(y)(i)}} \vdash s/(s\backslash np)/n \qquad \overset{\textit{linguist}}{\boxed{\mathsf{ling}}} \vdash n}{\overset{\textit{every linguist}}{\boxed{\lambda q, i.\forall y.\mathsf{ling}(y)(i) \to q(y)(i)}} \vdash s/(s\backslash np)} >$$

The resulting expression has the syntactic type of a quantifier; in this case, it takes on its right an expression which takes a noun phrase on its left to form a sentence, and it forms a sentence with that expression. This type—$s/(s\backslash np)$—is mirrored by the type of the λ-term which is the expression's semantic value: $(e \to \iota \to t) \to \iota \to t$. Indeed, the two are related by a *type homomorphism*; i.e., a map from syntactic types to semantic types that preserves certain structure—here, the structure of syntactic types formed via slashes (/ and \), which get turned into semantic types formed via arrows ($\to$). We refer to this type homomorphism as a "vanilla" type homomorphism to distinguish it from the monadic type homomorphisms which we introduce shortly.

Definition 3.5 (Vanilla type homomorphism)**.**

$$\begin{aligned} [\![b\backslash a]\!] &= [\![b/a]\!] \\ &= [\![a]\!] \to [\![b]\!] \end{aligned}$$

We may further codify the behavior of this homomorphism on atomic syntactic types (treating only *np*, *n*, and *s* for now).

(5) $[\![np]\!] = e$
$[\![n]\!] = e \to \iota \to t$
$[\![s]\!] = \iota \to t$

Indeed, the CCG derivation given in (4) tacitly assumes that noun phrases denote entities, that nouns denote functions from entities to *propositions* (i.e., functions of type $\iota \to t$), and that sentences denote propositions.

Crucially, *every* CCG rule is analogous to the application rules in that it preserves the structure of syntactic types in the types of semantic values via the type homomorphism. For another example, the rightward composition rule can be used to combine *every linguist* with *saw*.

[6]These conventions mirror each other in the sense that the input type of a function type is assumed to be atomic unless otherwise specified by the use of parentheses.

$$\begin{array}{ccc} \textit{Left identity} & \textit{Right identity} & \textit{Associativity} \\ v^{\eta} \star k \;=\; k(v) & m \star \lambda x.x^{\eta} \;=\; m & (m \star n) \star o \;=\; m \star \lambda x.n(x) \star o \end{array}$$

Figure 2: The monad laws.

(6)
$$\dfrac{\overset{\textit{every linguist}}{\boxed{\lambda q, i.\forall y.\mathsf{ling}(y)(i) \to q(y)(i)}} \vdash s/(s\backslash np) \qquad \overset{\textit{saw}}{\boxed{\mathsf{see}}} \vdash s\backslash np/np}{\overset{\textit{every linguist saw}}{\boxed{\lambda x, i.\forall y.\mathsf{ling}(y)(i) \to \mathsf{see}(x)(y)(i)}} \vdash s/np} > \mathbf{B}$$

Here, the resulting type—s/np—is mapped to $[\![np]\!] \to [\![s]\!] = e \to \iota \to t$, which is precisely the type of the resulting semantic value.

Likewise for the type-raising rules: in (7), *Jo*'s syntactic type is lifted from np to $s\backslash(s/np)$, while its semantic type is lifted from $[\![np]\!] = e$ to $[\![s\backslash(s/np)]\!] = (e \to \iota \to t) \to \iota \to t$.

(7)
$$\dfrac{\overset{\textit{every linguist saw}}{\boxed{\lambda x, i.\forall y.\mathsf{ling}(y)(i) \to \mathsf{see}(x)(y)(i)}} \vdash s/np \qquad \dfrac{\overset{\textit{Jo}}{\boxed{\mathsf{j}}} \vdash np}{\overset{\textit{Jo}}{\boxed{\lambda k.k(\mathsf{j})}} \vdash s\backslash(s/np)} < \mathbf{T}}{\overset{\textit{every linguist saw Jo}}{\boxed{\lambda x, i.\forall y.\mathsf{ling}(y)(i) \to \mathsf{see}(\mathsf{j})(y)(i)}} \vdash s} <$$

3.1.2 Monads

In monadic CCG, the vanilla type homomorphism is abstracted somewhat so that it may be defined in terms of an arbitrary *monad.*

Definition 3.6 (Monad)**.** A monad is a map $\mathtt{M}$ from types to types (i.e., given a semantic type α, $\mathtt{M}\alpha$ is some new semantic type), equipped with two operators, $\star$ and η, satisfying the monad laws (Figure 2).

$$(\cdot)^{\eta} : \alpha \to \mathtt{M}\alpha \qquad \text{(‘return’)}$$
$$(\star) : \mathtt{M}\alpha \to (\alpha \to \mathtt{M}\beta) \to \mathtt{M}\beta \qquad \text{(‘bind’)}$$

The leading intuition behind the map $\mathtt{M}$ is that if α is the type of a value, then $\mathtt{M}\alpha$ is the type of *programs* returning values of type α. The operator η turns values v of type α into programs which do nothing but return v, while $\star$ sequences one program that returns things of type α with a continuation that maps things of type α into programs which return things of type β. Reflecting this characterization, a monad must satisfy certain laws, provided in Figure 2. These laws ensure that lifting a value v of type α into the monad via η yields the kind of trivial program just described (the identity laws), as well as that the sequencing operation $\star$ is associative; i.e., in sequencing m, n, and o, it doesn't matter whether m is combined with n first or n is combined with o first.

The particular monad we employ as we develop PDS is the Reader.State.Set monad, which equips semantic values with a notion of dynamism (see Charlow 2014).

Definition 3.7 (The Reader.State.Set monad)**.**

$$\begin{aligned} \mathtt{M}\alpha &= \iota \to \gamma \to (\alpha \times \gamma) \to t \\ v^{\eta} &= \lambda i, g.\{\langle v, g\rangle\} \\ m \star k &= \lambda i, g. \bigcup_{\langle x, g'\rangle \in m(i)(g)} k(x)(i)(g') \end{aligned}$$

Thus a program of type $\mathtt{M}e$—one which "returns" entities—first reads in an index of type ι, then non-deterministically updates a *state*, which is a γ-type list of DRs. Meanwhile, it also non-deterministically returns state-sensitive entities, which are paired up with the updated states. Such a monad therefore encodes both a notion of intensionality—by allowing dynamic semantic values to depend on an index—and a notion of dynamism—by allowing, e.g., entities access to non-determinism and state-sensitivity, along with the ability to non-deterministically modify incoming states.

For example, the semantic value of *a linguist* can be encoded as such a program of type $\mathtt{M}e$.

(8) $\dfrac{\textit{a linguist}}{\boxed{\lambda i, g.\{\langle x, x{::}g\rangle \mid \mathsf{ling}(x)(i)\}}} \vdash np$

Given an index i and a state g, *a linguist* gives back a set of entity-state pairs, where each entity is a linguist at i, while each state is like g, but with the relevant linguist appended to it as a new DR.

As a monad, $\mathtt{M}$ imbues not just entities, but all types of values with these notions of intensionality and dynamism, allowing them to percolate up semantic derivations as expressions combine. As examples will make clear, the operators η and $\star$ are crucial for this purpose. Reflecting its general status as contributing trivial effects, η, as defined by Reader.State.Set, allows semantic values to read in indices only to throw them out, while returning a (state-relative) set containing a single value paired up with a state which it has left intact. $\star$, meanwhile, sequences programs by (a) combining their dependence on indices, and (b) non-deterministically sequencing the states they modify, while feeding the values returned by the first program as the inputs to the second.

There is substantial precedent for this encoding of dynamism. Indeed, similar notions of non-determinism are found in the earliest treatments of dynamic semantics (Kamp 1981; Heim 1982), while stateful non-determinism, as employed here, has precedents in Groenendijk and Stokhof 1991; Dekker 1994, i.a. Importantly, the particular monadic treatment we employ—in particular, as it is used to model indefinites—is pioneered by Charlow (2014), and takes slightly different forms in Charlow 2020b; Charlow 2020a. Charlow (2020b), especially, shows how non-determinism may be combined with Reader functionality so that intentionsionality and non-determinism can work together seamlessly in semantic derivations.

3.1.3 Combining CCG with monads

In order to blend a monadic semantics together with CCG, we need to generalize our vanilla type homomorphism to countenance a certain degree of monadic involvement. A monadic type homomorphisms accomplishes this.[7]

[7]If one considers CCG slashes to be analogous to λ-calculus arrows, the monadic type homomorphism defined here appears to follow the homomorphism accompanying the call-by-value translation of typed λ-calculus into a monad

Definition 3.8 (Monadic type homomorphism)**.**

$$\begin{aligned}[\![b\backslash a]\!]_{\mathtt{M}} &= [\![b/a]\!]_{\mathtt{M}} \\ &= [\![a]\!]_{\mathtt{M}} \longrightarrow \mathtt{M}[\![b]\!]_{\mathtt{M}}\end{aligned}$$

Thus syntactic types whose main connective is a slash are mapped to arrow types (as with the vanilla type homomorphism), but the result type is now prefixed by '$\mathtt{M}$', making it the kind of type an inhabitant of which can occur as the second argument of $\star$. Meanwhile, we may continue to codify the interpretation of atomic syntactic types as we did for the vanilla homomorphism (again, treating only np, n, and s for now—for $q_{ind.}$ and $q_{deg.}$, see (27) of Section 3.2.6; for $ap_{mod.}$, ap, and d, see (40) of Section 4.1; for sc, see (57) of Section 4.3).

(9) $[\![np]\!]_{\mathtt{M}} = e$
$[\![n]\!]_{\mathtt{M}} = e \longrightarrow \iota \longrightarrow t$
$[\![s]\!]_{\mathtt{M}} = \iota \longrightarrow t$

In general, if an expression has syntactic type c, we may assume that it has semantic type $\mathtt{M}[\![c]\!]_{\mathtt{M}}$. In fact, this situation is quite convenient. In the context of the Reader.State.Set monad, it allows indefinite noun phrases to be regarded as having type np, so that, e.g., *a linguist* may have the interpretation of type $\mathtt{M}e$ provided in (8) above. Meanwhile, indefinite determiners themselves may retain their usual type, np/n, as (10) illustrates.

(10) $$\overset{a}{\boxed{(\lambda p, i, g.\{\langle x, x{::}g\rangle \mid p(x)(i)\})^{\eta}}} \vdash np/n$$

This expression's semantic type can be read off of its syntactic type by following Definition 3.8.

$$\begin{aligned}\mathtt{M}[\![np/n]\!]_{\mathtt{M}} &= \mathtt{M}([\![n]\!]_{\mathtt{M}} \longrightarrow \mathtt{M}[\![np]\!]_{\mathtt{M}}) \\ &= \mathtt{M}((e \longrightarrow \iota \longrightarrow t) \longrightarrow \mathtt{M}[\![np]\!]_{\mathtt{M}}) \\ &= \mathtt{M}((e \longrightarrow \iota \longrightarrow t) \longrightarrow \mathtt{M}e)\end{aligned}$$

Taking $\mathtt{M}$ to be Reader.State.Set, one can check that this type is precisely that of the λ-term in (10), which is a function (returned by η) that maps a function from entities to propositions into monadic programs that return entities on which that function gives back true at the relevant index.

We provide the full set of monadic CCG rule schemata in Figure 3. These feature composition (**B**), type-raising (**T**), and substitution (**S**) rules. To illustrate their application, three derivations of the sentence *everyone ran a race* are given in Figure 4. Two of these derivations give the

(see Wadler 1992, §8 for the translation, as well as Shan 2004 for an application of the call-by-value translation to semantic types); meanwhile, the interpretation of monadic CCG rules given in Figure 3 follows the translation itself (though note that, with the exception of the type-raising rules, effects are discharged "as soon as possible", in a way that mirrors the order of the expressions). For that reason, the present system might be better regarded as "call-by-value" monadic CCG, in contrast to one that follows the call-by-name translation. We believe the call-by-value regime may be more suited to the exceptional scope properties of expressions whose interpretations monads are typically used to model (for key discussions of monads and exceptional scope, see Charlow 2014; Charlow 2020b; Charlow 2020a).

$$\dfrac{\overset{s_1}{\boxed{m_1}} \vdash c/b \qquad \overset{s_2}{\boxed{m_2}} \vdash b \mid_n a_n \;\cdots \mid_1 a_1}{\overset{s_1\, s_2}{\boxed{m_1 \star \lambda k.m_2 \star \lambda v.v^\eta \star \lambda u_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.u_n(x_n) \star k)^\eta \cdots)^\eta}} \vdash c \mid_n a_n \;\cdots \mid_1 a_1} > \mathbf{B}_n$$

$$\dfrac{\overset{s_1}{\boxed{m_1}} \vdash b \mid_n a_n \;\cdots \mid_1 a_1 \qquad \overset{s_1}{\boxed{m_2}} \vdash c\backslash b}{\overset{s_1\, s_2}{\boxed{m_1 \star \lambda v.m_2 \star \lambda k.v^\eta \star \lambda u_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.u_n(x_n) \star k)^\eta \cdots)^\eta}} \vdash c \mid_n a_n \;\cdots \mid_1 a_1} < \mathbf{B}_n$$

$$\dfrac{\overset{s}{\boxed{m}} \vdash a}{\overset{s}{\boxed{(\lambda k.(\lambda x_1.\cdots(\lambda x_n.m \star \lambda v.k(v) \star \lambda f_1.f_1(x_1) \star \cdots \star \lambda f_n.f_n(x_n))^\eta \cdots)^\eta)^\eta}} \vdash c \mid_n b_n \;\cdots \mid_1 b_1/(c \mid_n b_n \;\cdots \mid_1 b_1\backslash a)} > \mathbf{T}_n$$

$$\dfrac{\overset{s}{\boxed{m}} \vdash a}{\overset{s}{\boxed{(\lambda k.(\lambda x_1.\cdots(\lambda x_n.m \star \lambda v.k(v) \star \lambda f_1.f_1(x_1) \star \cdots \star \lambda f_n.f_n(x_n))^\eta \cdots)^\eta)^\eta}} \vdash c \mid_n b_n \;\cdots \mid_1 b_1\backslash(c \mid_n b_n \;\cdots \mid_1 b_1/a)} < \mathbf{T}_n$$

$$\dfrac{\overset{s_1}{\boxed{m_1}} \vdash c/b \mid_n a_n \;\cdots \mid_1 a_1 \qquad \overset{s_2}{\boxed{m_2}} \vdash b \mid_n a_n \;\cdots \mid_1 a_1}{\overset{s_1\, s_2}{\boxed{m_1 \star \lambda k.m_2 \star \lambda v.k^\eta \star \lambda f_1.v^\eta \star \lambda u_1.(\lambda x_1.f_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.f_n(x_n) \star \lambda g.u_n(x_n) \star \lambda w.w \star g)^\eta \cdots)^\eta}} \vdash c \mid_n a_n \;\cdots \mid_1 a_1} > \mathbf{S}_n$$

$$\dfrac{\overset{s_1}{\boxed{m_1}} \vdash b \mid_n a_n \;\cdots \mid_1 a_1 \qquad \overset{s_2}{\boxed{m_2}} \vdash c\backslash b \mid_n a_n \;\cdots \mid_1 a_1 \cdots \mid_1 a_1}{\overset{s_1\, s_2}{\boxed{m_1 \star \lambda v.m_2 \star \lambda k.v^\eta \star \lambda u_1.k^\eta \star \lambda f_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda f_n.(\lambda x_n.u_n(x_n) \star \lambda w.f_n(x_n) \star \lambda g.w \star g)^\eta \cdots)^\eta}} \vdash c \mid_n a_n \;\cdots \mid_1 a_1} < \mathbf{S}_n$$

Figure 3: Monadic CCG rule schemata.

indefinite noun phrase wide scope over *everyone*—though via different mechanisms—and the third derivation gives the subject quantifier wide scope. Note that the rule schemata define n^{th} order instances of each rules type. Intuitively, n is the number of arguments which need to be "gotten out of the way" when applying rules. Thus for instance, $\mathbf{B}_0$ and $\mathbf{S}_0$ coincide and correspond to the standard application rules.

$$\frac{\overset{s_1}{\boxed{m_1}} \vdash c/b \qquad \overset{s_2}{\boxed{m_2}} \vdash b}{\overset{s_1\ s_2}{\boxed{m_1 \star \lambda k.m_2 \star \lambda v.k(v)}} \vdash c} > \qquad \frac{\overset{s_1}{\boxed{m_1}} \vdash b \qquad \overset{s_2}{\boxed{m_2}} \vdash c\backslash b}{\overset{s_1\ s_2}{\boxed{m_1 \star \lambda v.m_2 \star \lambda k.k(v)}} \vdash c} <$$

Likewise, the standard **B** rules may be identified with the $\mathbf{B}_1$ rules, and similarly for the **S** rules. Meanwhile, type-raising may now be seen as somewhat analogous to Charlow's (2014) employment of the continuation monad to lift indefinite semantic values into quantifier semantic values. Note that the analogy is especially apparent in the case of 0th-order type-raising, which effectively provides the resulting quantifier with widest scope.

$$\frac{\overset{s}{\boxed{m}} \vdash a}{\overset{s}{\boxed{(\lambda k.m \star k)^\eta}} \vdash c/(c\backslash a)} >\mathbf{T}_0 \qquad \frac{\overset{s}{\boxed{m}} \vdash a}{\overset{s}{\boxed{(\lambda k.m \star k)^\eta}} \vdash c\backslash(c/a)} <\mathbf{T}_0$$

To close out, we note a couple of important properties that monadic CCG enjoys. First, since we can in principle pick *any* monad to be $\mathtt{M}$, one possibility is to pick the Identity monad, which does nothing to types or terms.

Definition 3.9 (The Identity monad)**.**

$$\begin{aligned} \mathtt{M}\alpha &= \alpha \\ v^\eta &= v \\ m \star k &= k(m) \end{aligned}$$

By considering the rules in Figure 3, one can see that choosing $\mathtt{M}$ to be Identity gives back vanilla CCG.[8] It is in this sense that monadic CCG is a proper *generalization* of CCG.

Second, there is an important sense in which monadic CCG forms a coherent system, insofar as expressions can be combined using the rules in Figure 3 in any way, any number of times, and the monadic type homomorphism will be preserved. A few definitions are helpful to flesh this idea out. First, there is sense in which individual expressions can be seen to be coherent in view of some monad $\mathtt{M}$.

Definition 3.10 (Well-typed by $\mathtt{M}$)**.** An expression $\overset{s}{\boxed{m}} \vdash c$ is well-typed by $\mathtt{M}$ iff $m : \mathtt{M}[\![c]\!]_{\mathtt{M}}$.

That is, an expression is coherent if its syntactic and semantic types are in harmony. Second, we can also view a lexicon as being a set of such expressions.

[8]This choice effectively collapses the type-raising rules into two—the standard >**T** and <**T** rules—since orders of type-raising higher than 0 now amount to η-expansions.

Definition 3.11 (M-CCG lexicon)**.** An M-CCG lexicon is a finite set of expressions well-typed by M.

Finally, we have the following theorem.

Theorem 3.12 (Preservation of well-typedness)**.** For any monad M, if an expression *e* is derivable from an M-CCG lexicon by the rule schemata in Figure 3, then *e* is well-typed by M.

Proof. By simultaneous induction on the size of rules and the height of derivations. □

Because of Theorem 3.12, we can effectively plug and play with any monad M under the sun, and we will be able to read the semantic type of an expression derived from an M-CCG lexicon off of its syntactic type. This means that, insofar as monads equip our grammars with sufficient expressiveness to encode the kinds of effects that interest us, the types of these effects were—in a way—always baked into CCG! A version of this property also carries over to proposals made in Section 3.2—e.g., those regarding the probabilistic variant of monadic CCG. These proposals will be made in as general a format as possible, in order to countenance the possibility that one might wish to use the present system without its particular commitment to Reader.State.Set.

1.

$$
\dfrac{
\overset{\textit{everyone}}{\boxed{(\lambda k,\_,g.\{\langle \lambda i.\forall y.\mathsf{person}(y)(i) \rightarrow \exists\langle p,g'\rangle \in k(y)(i)(g) : p(i), g\rangle\})^{\eta}}} \vdash s/(s\backslash np)
\quad
\dfrac{
\overset{\textit{ran}}{\boxed{(\lambda x.(\lambda y.\mathsf{run}(x)(y)^{\eta})^{\eta})^{\eta}}} \vdash s\backslash np/np
\quad
\dfrac{
\overset{\textit{a}}{\boxed{(\lambda p,i,g.\{\langle x, x{::}g\rangle \mid p(x)(i)\})^{\eta}}} \vdash np/n
\quad
\overset{\textit{race}}{\boxed{\mathsf{race}^{\eta}}} \vdash n
}{
\overset{\textit{a race}}{\boxed{\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash np
}\,{>}
}{
\overset{\textit{ran a race}}{\boxed{\lambda i,g.\{\langle \lambda y.\mathsf{run}(x)(y)^{\eta}, x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash s\backslash np
}\,{>}
}{
\overset{\textit{everyone ran a race}}{\boxed{\lambda i,g.\{\langle \lambda i'.\forall y.\mathsf{person}(y)(i') \rightarrow \mathsf{run}(x)(y)(i'), x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash s
}\,{>}
$$

2.

$$
\dfrac{
\dfrac{
\overset{\textit{everyone}}{\boxed{(\lambda k,\_,g.\{\langle \lambda i.\forall y.\mathsf{person}(y)(i) \rightarrow \exists\langle p,g'\rangle \in k(y)(i)(g) : p(i), g\rangle\})^{\eta}}} \vdash s/(s\backslash np)
\quad
\overset{\textit{ran}}{\boxed{(\lambda x.(\lambda y.\mathsf{run}(x)(y)^{\eta})^{\eta})^{\eta}}} \vdash s\backslash np/np
}{
\overset{\textit{everyone ran}}{\boxed{(\lambda x,\_,g.\{\langle \lambda i.\forall y.\mathsf{person}(y)(i) \rightarrow \mathsf{run}(x)(y)(i), g\rangle\})^{\eta}}} \vdash s/np
}\,{>}\mathbf{B}
\quad
\dfrac{
\dfrac{
\overset{\textit{a}}{\boxed{(\lambda p,i,g.\{\langle x, x{::}g\rangle \mid p(x)(i)\})^{\eta}}} \vdash np/n
\quad
\overset{\textit{race}}{\boxed{\mathsf{race}^{\eta}}} \vdash n
}{
\overset{\textit{a race}}{\boxed{\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash np
}\,{>}
}{
\overset{\textit{a race}}{\boxed{(\lambda k.(\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{race}(x)(i)\}) \star k)^{\eta}}} \vdash s\backslash(s/np)
}\,{<}\mathbf{T}_0
}{
\overset{\textit{everyone ran a race}}{\boxed{\lambda i,g.\{\langle \lambda i'.\forall y.\mathsf{person}(y)(i') \rightarrow \mathsf{run}(x)(y)(i'), x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash s
}\,{<}
$$

3.

$$
\dfrac{
\overset{\textit{everyone}}{\boxed{(\lambda k,\_,g.\{\langle \lambda i.\forall y.\mathsf{person}(y)(i) \rightarrow \exists\langle p,g'\rangle \in k(y)(i)(g) : p(i), g\rangle\})^{\eta}}} \vdash s/(s\backslash np)
\quad
\dfrac{
\overset{\textit{ran}}{\boxed{(\lambda x.(\lambda y.\mathsf{run}(x)(y)^{\eta})^{\eta})^{\eta}}} \vdash s\backslash np/np
\quad
\dfrac{
\dfrac{
\overset{\textit{a}}{\boxed{(\lambda p,i,g.\{\langle x, x{::}g\rangle \mid p(x)(i)\})^{\eta}}} \vdash np/n
\quad
\overset{\textit{race}}{\boxed{\mathsf{race}^{\eta}}} \vdash n
}{
\overset{\textit{a race}}{\boxed{\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{race}(x)(i)\}}} \vdash np
}\,{>}
}{
\overset{\textit{a race}}{\boxed{(\lambda k.(\lambda y.(\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{race}(x)(i)\}) \star \lambda v.k(v) \star \lambda f.f(y))^{\eta})^{\eta}}} \vdash s\backslash np\backslash(s\backslash np/np)
}\,{<}\mathbf{T}_1
}{
\overset{\textit{ran a race}}{\boxed{(\lambda y,i,g.\{\langle \mathsf{run}(x)(y), x{::}g\rangle \mid \mathsf{race}(x)(i)\})^{\eta}}} \vdash s\backslash np
}\,{<}
}{
\overset{\textit{everyone ran a race}}{\boxed{(\lambda i.\forall y.\mathsf{person}(y)(i) \rightarrow \exists x.\mathsf{race}(x)(i) \wedge \mathsf{run}(x)(y)(i))^{\eta}}} \vdash s
}\,{>}
$$

Figure 4: Three analyses of the sentence *everyone ran a race*. 1 and 2 give the indefinite noun phrase wide scope: 1, by monad-aware composition, and 2, by first lifting it into a quantifier and then invoking the >**B** rule. 3 gives the indefinite narrow scope by lifting it into a quantifier and using the application rules.

$$\frac{\Gamma \vdash t : \alpha}{\Gamma \vdash \boxed{t} : \mathsf{P}\alpha}\,\texttt{Return} \qquad\qquad \frac{\Gamma \vdash t : \mathsf{P}\alpha \qquad \Gamma, x : \alpha \vdash u : \mathsf{P}\beta}{\Gamma \vdash \begin{pmatrix} x \sim t \\ u \end{pmatrix} : \mathsf{P}\beta}\,\texttt{Bind}$$

Figure 5: Probabilistic programs.

3.2 Probabilistic dynamism

With the preliminaries now in place, we turn to formalizing the probabilistic components of PDS.

3.2.1 Augmenting the type system

The type system presented in Section 3.1 included types for entities, truth values, lists of entities, indices, and complex types formed from these. Here, we follow Grove and Bernardy (2023), who show how a semantics incorporating Bayesian reasoning can be encoded using a λ-calculus with such a type system; however, whereas Grove and Bernardy represent probabilistic reasoning using continuations, we employ a somewhat more abstract presentation by incorporating a new type constructor (P). New atomic types are given in Definition 3.13.

Definition 3.13 (Atomic types)**.**

$$A ::= e \mid t \mid \gamma \mid r$$

The full (and final) set of types is given in Definition 3.14.

Definition 3.14 (Types over A)**.**

$$\mathcal{T}_A ::= A \mid \mathcal{T}_A \rightarrow \mathcal{T}_A \mid \mathcal{T}_A \times \mathcal{T}_A \mid \diamond \mid \mathsf{P}\mathcal{T}_A$$

Types of the form $\mathsf{P}\alpha$ are inhabited by *probabilistic programs* that represent probability distributions over values of type α. For example, a program of type $\mathsf{P}t$ represents a probability distribution over truth values (i.e., a Bernoulli distribution); a program of type $\mathsf{P}e$ represents a probability distribution over entities (e.g., a categorical distribution); and a program of type $\mathsf{P}(e \rightarrow t)$ represents a probability distribution over functions from entities to truth values.

There are two important things to note about Definition 3.14. First, we have gotten rid of the type of indices ι, which we will reintegrate by other means in Section 3.2.2. Second, we have added a new atomic type r, which is the type of real numbers. This type is useful for specifying semantic values that make use of *degrees*—e.g., the semantic value of *tall*—as well as for modeling probabilities. For example, a program of type $\mathsf{P}r$ represents a probability distribution over real numbers—e.g., a normal distribution.

The probability monad. Importantly, the type constructor P is also defined to be a monad. The syntax of programs of type $\mathsf{P}\alpha$ is defined in Figure 5, which features one rule corresponding to each monadic operator: `Return` (an introduction rule) and `Bind` (an elimination rule), which provide constructors allowing for definitions of *return* and *bind*.[9] As a monad, P (together with

[9]See Bernardy, Blanck, Chatzikyriakidis, and Maskharashvili 2022, where similar rules are presented in a somewhat more refined, dependently typed setting.

return and bind) should satisfy the monad laws in Figure 2—a result guaranteed by the equalities in (11) (where each is labeled according to the law it supports).[10]

(11) *Left identity* *Right identity* *Associativity*

$$\begin{array}{l} x \sim \boxed{v} \\ k \end{array} = k[v/x] \qquad \begin{array}{l} x \sim m \\ \boxed{x} \end{array} = m \qquad \begin{array}{l} y \sim \begin{pmatrix} x \sim m \\ n \end{pmatrix} \\ o \end{array} = \begin{array}{l} x \sim m \\ y \sim n \\ o \end{array}$$

Our motivation for presenting bind using the '$\sim$' notation is to evoke *distribution* (or *sampling*) *statements*, as employed in probability theory. That bind can be conceptualized in terms of sampling allows us to understand probabilistic programs of type $\mathsf{P}\alpha$ (granted, somewhat metaphorically) as though they are *computing* random values of type α.

Our first probabilistic program. We can recruit return and bind to characterize complex probability distributions. To illustrate, suppose we have some categorical distribution, $\mathsf{mammal} : \mathsf{P}e$, on mammals. We can represent a distribution on mammals' mothers as in (12).

(12) $$\begin{array}{l} x \sim \mathsf{mammal} \\ \boxed{\mathsf{mother}(x)} \end{array}$$

Here, a random entity $x : e$ is *sampled* from $\mathsf{mammal} : \mathsf{P}e$ using bind, and then $\mathsf{mother}(x) : e$ is returned, as indicated by the orange box. Since return turns things of type α into probabilistic programs of type $\mathsf{P}\alpha$, the resulting probabilistic program is of type $\mathsf{P}e$. Furthermore, assuming that the probability distribution mammal only has support on (i.e., assigns non-zero probability to) the entities which are mammals, the distribution which results will only have support on the entities which are the mothers of entities which are mammals.

[10]Going from left to right, one can view such equalities as giving rise to *reduction rules* (Benton, Bierman, and Paiva 1998). In particular, Left identity may be viewed as β-reduction ($\Rightarrow_\beta$), Right identity, as η-reduction ($\Rightarrow_\eta$), and Associativity, meanwhile, as a commuting conversion ($\Rightarrow_c$).

Because we take the constructors for probabilistic programs to be primitives of the language, such reduction rules are not redundant with those of the ambient λ-calculus. Moreover, it is useful to have both Left identity and Associativity, since they allow probabilistic programs to be reduced to a normal form in which any return statement only occurs either at the end of a program or inside the argument of a function. Consider the following reduction, for instance, in which the returned u, otherwise trapped, is eliminated in favor of a direct substitution:

$$\begin{array}{ll} & \begin{array}{l} y \sim \begin{pmatrix} x \sim t \\ \boxed{u} \end{pmatrix} \\ v \end{array} \\ \Rightarrow_c & \begin{array}{l} x \sim t \\ y \sim \boxed{u} \\ v \end{array} \\ \Rightarrow_\beta & \begin{array}{l} x \sim t \\ v[u/y] \end{array} \end{array}$$

Reweighting distributions. Our probabilistic language also comes with a function $\texttt{factor}$ for scaling probability distributions according to some weight.[11]

(13) $\texttt{factor} : r \longrightarrow \mathsf{P}\diamond$

For instance, we may constrain our "mother" distribution so that it assigns more weight to the mothers of mammals which are hungrier.

(14) $x \sim \mathsf{mammal}$
$\texttt{factor}(\mathsf{hungry}(x))$
$\boxed{\mathsf{mother}(x)}$

Here, $\mathsf{hungry} : e \longrightarrow r$ maps entities onto degrees representing how hungry they are. Thus the program above represents a probability distribution over entities which assigns non-zero probabilities only to entities which are the mother of some mammal, and which assigns greater probabilities to entities the hungrier their children are.

Making observations. In terms of $\texttt{factor}$, we may define another function, $\mathsf{observe}$.

Definition 3.15 ($\mathsf{observe}$).

$$\begin{aligned} \mathsf{observe} &: t \longrightarrow \mathsf{P}\diamond \\ \mathsf{observe}(p) &= \texttt{factor}(\mathbb{1}(p)) \end{aligned}$$

$\mathsf{observe}$ takes a truth value and either keeps or throws out the distribution represented by the expression which follows it, depending on whether this truth value is T or F. This is accomplished by factoring a distribution by the value of an indicator function ($\mathbb{1}$) applied to the truth value.[12]

Definition 3.16 (Indicator function).

$$\begin{aligned} \mathbb{1} &: t \longrightarrow r \\ \mathbb{1}(\mathsf{T}) &= 1 \\ \mathbb{1}(\mathsf{F}) &= 0 \end{aligned}$$

For instance, we may instead constrain our "mother" program to describe a distribution over only dogs' mothers.

(15) $x \sim \mathsf{mammal}$
$\mathsf{observe}(\mathsf{dog}(x))$
$\boxed{\mathsf{mother}(x)}$

[11] We define $\texttt{factor}$ here as a primitive of the language of probabilistic programs (i.e., a constant). In their continuation-based treatment, Grove and Bernardy (2023) implement $\texttt{factor}$ so that it has the scaling behavior described only informally here. One could (if they wanted to) interpret the current system into one that uses continuations, so that $\texttt{factor}$ has the behavior needed.

[12] See Grove and Bernardy 2023 for further details.

This distribution assigns a probability of 0 to any entity which is not the mother of some dog. Indeed, we could use both `factor` and `observe` to define another distribution which assigns a probability of 0 to any entity which is not the mother of some dog, and which assigns greater probabilities to mothers of hungrier dogs.

(16) $x \sim \mathsf{mammal}$
$\mathtt{factor}(\mathsf{hungry}(x))$
$\mathtt{observe}(\mathsf{dog}(x))$
$\boxed{\mathsf{mother}(x)}$

3.2.2 The common ground

Here, we make good on the assumption mentioned in Section 2.3 that common grounds amount to probability distributions over indices which consist of a possible world together with a context. In particular, we view the *type* of an index as being determined by a string of types, i.e., an element of $\mathcal{T}_A^*$ (the free monoid over $\mathcal{T}_A$). Given any such string u, there is an obvious map $\langle\!\langle\cdot\rangle\!\rangle : \mathcal{T}_A^* \to \mathcal{T}_A$ taking u onto an n-ary product type.

Definition 3.17 ($\langle\!\langle\cdot\rangle\!\rangle$)**.**

$$\langle\!\langle\epsilon\rangle\!\rangle = \diamond$$
$$\langle\!\langle au\rangle\!\rangle = a \times \langle\!\langle u\rangle\!\rangle$$

Indices may then be regarded as tuples "inhabiting" a particular string of types, which we take to be a member of a fixed set $\mathcal{S}_\iota$; i.e., any index i is such that $i : \langle\!\langle\iota\rangle\!\rangle$ for some $\iota \in \mathcal{S}_\iota$—in this case, we will simply write '$i : \iota$' when it doesn't introduce confusion. Importantly, there is nothing deep about our use of *strings* of types. They are mainly here to simplify notation.

Definition 3.18 (World-context parameters)**.** The set of world-context parameters is a set $\mathcal{S}_\iota \subseteq \mathcal{T}_A^*$ together with a preorder $\lesssim_\iota$ on $\mathcal{S}_\iota$.

For example, if we think of an index i as determining various entities' heights, then $i : u\,(e \to r)\,v$; i.e., i has some $(|u|+1)^{\text{th}}$ projection $\pi_{|u|+1}(i) : e \to r$ mapping entities to real numbers representing their heights. As a matter of convention, we simply write '$\mathsf{height}(i)$' in this case. The preorder $\lesssim_\iota$ encodes when one parameter provides at most all of the same projections as another; i.e., $\iota \lesssim_\iota \iota'$ if for any $i : \iota$ and $i' : \iota'$, any "constant" c (e.g., height) is defined on i' if it is defined on i. It is thus the preorder itself which determines when projections of indices of different types correspond to the same "constant".

The following definitions allow us to pull separate world parts and context parts out of a given parameter, as well as worlds and contexts out of a given index.

Definition 3.19 (Worlds and contexts)**.** There are functions

$$\omega : \mathcal{S}_\iota \to \mathcal{T}_A^*$$
$$\kappa : \mathcal{S}_\iota \to \mathcal{T}_A^*$$

taking any world-context parameter onto its world part and context part, respectively. Furthermore, given a world context-parameter ι, there are functions

$$\begin{aligned} w &: \iota \longrightarrow \omega_\iota \\ c &: \iota \longrightarrow \kappa_\iota \\ i &: \omega_\iota \times \kappa_\iota \longrightarrow \iota \end{aligned}$$

where w and c take any given index i onto its associated world w_i and context c_i, respectively, while i takes a world and a context onto their associated index. Indeed, these functions satisfy the laws for products.

$$\begin{aligned} w_{i_{w',\kappa'}} &= w' \\ \kappa_{i_{w',\kappa'}} &= \kappa' \\ i_{w_{i'},\kappa_{i'}} &= i' \end{aligned}$$

Thus for example, given an index i, we may sometimes write '$\mathsf{height}(w_i)$' to indicate that the relevant projection of i of type $e \rightarrow r$ comes from the world part of i. Meanwhile, we may write, e.g., '$\mathsf{d}_{tall}(c_i)$' to denote the relevant projection of type r providing the height threshold for the gradable adjective *tall*.

Finally, we take *common grounds* to be of type $\mathsf{P}\iota$ for some $\iota \in \mathcal{S}_\iota$; i.e., they encode probability distributions over indices. Thus common grounds track how probable indices are, analogous to much previous work in probabilistic semantics and pragmatics (see Erk 2022).

3.2.3 Expressions and discourses

We define discourse states as lists of *metalinguistic parameters*, including, e.g., the common ground and the QUD, along with other conversationally relevant features of discourse (e.g., representations of the entities to which pronouns can refer, the available antecedents for ellipsis, etc.). One can view a discourse state as akin to the context state of Farkas and Bruce 2010, though the type of state we employ is in principle less constrained, insofar as the type of individual parameters is open ended.

Definition 3.20 (State parameters)**.** The set of state parameters is a set $\mathcal{S}_\sigma \subseteq \mathcal{T}_A^*$ together with a preorder $\lesssim_\sigma$ on $\mathcal{S}_\sigma$.

We assume notational conventions for state parameters analogous to those we defined for world-context parameters. For example, we sometimes write '$s : \sigma$' in place of '$s : \langle\!\langle\sigma\rangle\!\rangle$' when it doesn't introduce confusion (e.g., outside of formal definitions). Meanwhile, we obtain the projections of states in terms of "constants", and the preorder $\leq_\sigma$ serves an analogous function to that of $\leq_\iota$; e.g., for a given state s, we write '$\mathtt{CG}(s)$' to obtain the common ground, as well as '$\mathtt{QUD}(s)$' to obtain the QUD stack.

We regard expressions' probabilistic semantic values as functions of type $\langle\!\langle\sigma\rangle\!\rangle \rightarrow \mathtt{P}(\mathtt{M}\alpha \times \langle\!\langle\sigma'\rangle\!\rangle)$, where $\sigma, \sigma' \in \mathcal{S}_\sigma$ and $\mathtt{M}\alpha$ is an expression's ordinary (dynamic) semantic type. We abbreviate these types $\mathbb{P}^{\sigma}_{\sigma'}(\mathtt{M}\alpha)$. In principle, $\mathtt{M}$ may be any monad, though here we make the concrete choice to identify it with the Reader.State.Set monad defined in Section 3.1.2. Thus given an input state $s : \sigma$, the semantic value of an expression produces a probability distribution over pairs of

Left identity *Right identity*

$$\begin{array}{l}\mathsf{do}_{p,p,q}\; x \leftarrow \boxed{v}_p \\ \quad k(x)\end{array} \;=\; k(v) \qquad\qquad \begin{array}{l}\mathsf{do}_{p,q,q}\; x \leftarrow m \\ \quad \boxed{x}_q\end{array} \;=\; m$$

Associativity

$$\begin{array}{l}\mathsf{do}_{p,r,s}\; y \leftarrow \left(\begin{array}{l}\mathsf{do}_{p,q,r}\; x \leftarrow m \\ \quad n(x)\end{array}\right) \\ \quad o(y)\end{array} \;=\; \begin{array}{l}\mathsf{do}_{p,q,s}\; x \leftarrow m \\ \left(\begin{array}{l}\mathsf{do}_{q,r,s}\; y \leftarrow n(x) \\ \quad o(y)\end{array}\right)\end{array}$$

Figure 6: The parameterized monad laws.

ordinary dynamic semantic values of type $\mathsf{M}\alpha$ and possible output states $s' : \sigma'$. An expression of category *np*, for instance, now has the *probabilistic* type $\mathbb{P}^{\sigma}_{\sigma'}(\mathsf{M}[\![np]\!]_{\mathsf{M}}) = \mathbb{P}^{\sigma}_{\sigma'}(\mathsf{M}e)$.[13]

Building on this view of expressions, we regard an ongoing discourse as a function of type $\mathbb{P}^{\sigma}_{\sigma'}\diamond$. The effects that both expressions and discourses have are *stateful-probabilistic*: they map input states to probability distributions over output states. Discourses differ from expressions in that the value discourses compute is trivial: it is invariably the empty tuple $\diamond$, as determined by its type. Thus while expressions produce both stateful-probabilistic effects *and* values, discourses have *only* effects, i.e., they merely update the state.

3.2.4 Semantic composition via parameterized monads

The setup we have introduced allows for the possibility that the state parameter σ *changes* in the course of evaluating an expression's probabilistic semantic value. Such a value may map an input state $s : \sigma$ onto a probability distribution over outputs states of type σ' ($\neq \sigma$). This flexibility is necessary to capture the changing nature of certain components of the discourse state. For example, the QUD stack may have arbitrarily many questions on it, and those questions may themselves have different types—e.g., degree questions, individual questions, etc. Thus whenever an utterance functions to add a question to the QUD stack, the input state's type will not match the output state's type.

To countenance this type-level flexibility, we view the types $\mathbb{P}^{\sigma}_{\sigma'}\alpha$ as arising from a *parameterized* State.Probability monad, given the collection $\mathcal{S}_\sigma$ of possible state parameters.[14] Parameterized monads are associated with their own definitions of (parameterized) return and bind. To increase clarity, while distinguishing the notations for parameterized and vanilla monads, we present the bind statements of a parameterized monad $\mathbb{P}$ using Haskell's do-notation.

Definition 3.21 (Parameterized monad (Atkey 2009))**.** Given a collection $\mathcal{S}$ of parameters, a parameterized monad is a map $\mathbb{P}$ from triples consisting of two parameters and a type onto types (i.e., given parameters $p, q \in \mathcal{S}$ and a type α, $\mathbb{P}^{p}_{q}\alpha$ is some new type), equipped with two operators

[13]We continue to assume that a string of category *s* expresses a proposition, but we ambiguously identify the type ι of intensional indices with one of the types in $\mathcal{S}_\iota$, so that the probabilistic type of a sentence is one of $\{\mathbb{P}^{\sigma}_{\sigma'}(\mathsf{M}(\iota \to t))\}_{\iota \in \mathcal{S}_\iota}$. We elaborate on this more intricate view of semantic composition in Section 3.2.4.

[14]See Atkey 2009 on the parameterized State monad and parameterized monads more generally. The current parameterized monad can be viewed as applying a parameterized State monad *transformer* to the underlying probability monad P; see Liang, Hudak, and Jones 1995 on monad transformers.

satisfying the parameterized monad laws (Figure 6).

$$\begin{aligned} \boxed{(\cdot)}_p &: \alpha \to \mathbb{P}^p_p\alpha && (\text{`return'}) \\ \begin{array}{r}\mathsf{do}_{p,q,r}\ x \leftarrow \_\_ \\ \_\_(x)\end{array} &: \mathbb{P}^p_q\alpha \to (\alpha \to \mathbb{P}^q_r\beta) \to \mathbb{P}^p_r\beta && (\text{`bind'}) \end{aligned}$$

The particular parameterized monad we employ is State.Probability, where the relevant collection of parameters is $\mathcal{S}_\sigma$.

Definition 3.22 (The parameterized State.Probability monad)**.**

$$\begin{aligned} \mathbb{P}^\sigma_{\sigma'}\alpha &= \langle\!\langle\sigma\rangle\!\rangle \to \mathsf{P}(\alpha \times \langle\!\langle\sigma'\rangle\!\rangle) \\ \boxed{v}_\sigma &= \lambda s.\boxed{\langle v, s\rangle} \\ \begin{array}{r}\mathsf{do}_{\sigma,\sigma',\sigma''}\ x \leftarrow m \\ k(x)\end{array} &= \lambda s.\begin{pmatrix}\langle x, s'\rangle \sim m(s) \\ k(x)(s')\end{pmatrix} \end{aligned}$$

The do-notation in the above should be read as saying, "first bind the variable x to the program m, and then do $k(x)$". Indeed, this statement gives an intuitive summary of what the definition of State.Probability accomplishes: to bind m to the continuation k, one must abstract over an input state s and feed it to m, sample a value x paired with an output state s' from the result, and finally, feed x, along with s', to k.

We follow a couple of simplifying notational conventions throughout. First, we will generally leave the parameters on operators implicit, writing '$\boxed{x}$' instead of '$\boxed{x}_p$' and 'do' instead of '$\mathsf{do}_{p,q,r}$'. Second, we will follow Haskell's convention of representing multiple uses of bind in a row

$$\begin{array}{l}\mathsf{do}\ x \leftarrow m \\ \quad \mathsf{do}\ y \leftarrow n \\ \quad\quad k(x,y)\end{array}$$

by consolidating them into a single do-block,

$$\begin{array}{l}\mathsf{do}\ x \leftarrow m \\ \quad\ \ y \leftarrow n \\ \quad\ \ k(x,y)\end{array}$$

or by separating them on a single line by a semicolon to save space.

$$\mathsf{do}\ x \leftarrow m; y \leftarrow n; k(x,y)$$

The updated, *probabilistic* monadic CCG rule schemata are provided in Figure 7. These schemata mimic the ones defined in Section 3.1.3, except that semantic values are considered to be of type $\mathbb{P}^\sigma_{\sigma'}(\mathsf{M}\alpha)$ (for $\sigma, \sigma' \in \mathcal{S}_\sigma$) now, rather than of type $\mathsf{M}\alpha$. Thus rather than apply the monadic CCG operations to semantic values directly, we must bind these semantic values to variables of type $\mathsf{M}\alpha$ and apply the operations to *those.* Meanwhile, the probabilistic type associated with a given expression is related to its semantic type by allowing the parameters σ and σ' to range over $\mathcal{S}_\sigma$.

The upshot is that, while an expression's syntactic type continues to determine its dynamic semantic effects, its *probabilistic* dynamic effects are independent, lending them a certain amount of modularity.

$$\frac{\overset{s_1}{\boxed{M_1}} \vdash c/b \qquad \overset{s_2}{\boxed{M_2}} \vdash b \mid_n a_n \ \cdots \mid_1 a_1}{\overset{s_1\ s_2}{\boxed{\begin{array}{l}\mathsf{do}\ m_1 \leftarrow M_1; m_2 \leftarrow M_2\\ \quad\boxed{m_1 \star \lambda k.m_2 \star \lambda v.v^\eta \star \lambda u_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.u_n(x_n) \star k)^\eta \cdots)^\eta}\end{array}}} \vdash c \mid_n a_n \ \cdots \mid_1 a_1}\ {>}\mathbf{B}_n$$

$$\frac{\overset{s_1}{\boxed{M_1}} \vdash b \mid_n a_n \ \cdots \mid_1 a_1 \qquad \overset{s_1}{\boxed{M_2}} \vdash c\backslash b}{\overset{s_1\ s_2}{\boxed{\begin{array}{l}\mathsf{do}\ m_1 \leftarrow M_1; m_2 \leftarrow M_2\\ \quad\boxed{m_1 \star \lambda v.m_2 \star \lambda k.v^\eta \star \lambda u_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.u_n(x_n) \star k)^\eta \cdots)^\eta}\end{array}}} \vdash c \mid_n a_n \ \cdots \mid_1 a_1}\ {<}\mathbf{B}_n$$

$$\frac{\overset{s}{\boxed{M}} \vdash a}{\overset{s}{\boxed{\begin{array}{l}\mathsf{do}\ m \leftarrow M\\ \quad\boxed{(\lambda k.(\lambda x_1.\cdots(\lambda x_n.m \star \lambda v.k(v) \star \lambda f_1.f_1(x_1) \star \cdots \star \lambda f_n.f_n(x_n))^\eta \cdots)^\eta)^\eta}\end{array}}} \vdash c \mid_n b_n \ \cdots \mid_1 b_1/(c \mid_n b_n \ \cdots \mid_1 b_1\backslash a)}\ {>}\mathbf{T}_n$$

$$\frac{\overset{s}{\boxed{M}} \vdash a}{\overset{s}{\boxed{\begin{array}{l}\mathsf{do}\ m \leftarrow M\\ \quad\boxed{(\lambda k.(\lambda x_1.\cdots(\lambda x_n.m \star \lambda v.k(v) \star \lambda f_1.f_1(x_1) \star \cdots \star \lambda f_n.f_n(x_n))^\eta \cdots)^\eta)^\eta}\end{array}}} \vdash c \mid_n b_n \ \cdots \mid_1 b_1\backslash(c \mid_n b_n \ \cdots \mid_1 b_1/a)}\ {<}\mathbf{T}_n$$

$$\frac{\overset{s_1}{\boxed{M_1}} \vdash c/b \mid_n a_n \ \cdots \mid_1 a_1 \qquad \overset{s_2}{\boxed{M_2}} \vdash b \mid_n a_n \ \cdots \mid_1 a_1}{\overset{s_1\ s_2}{\boxed{\begin{array}{l}\mathsf{do}\ m_1 \leftarrow M_1; m_2 \leftarrow M_2\\ \quad\boxed{m_1 \star \lambda k.m_2 \star \lambda v.k^\eta \star \lambda f_1.v^\eta \star \lambda u_1.(\lambda x_1.f_1(x_1) \star \cdots \star \lambda u_n.(\lambda x_n.f_n(x_n) \star \lambda g.u_n(x_n) \star \lambda w.w \star g)^\eta \cdots)^\eta}\end{array}}} \vdash c \mid_n a_n \ \cdots \mid_1 a_1}\ {>}\mathbf{S}_n$$

$$\frac{\overset{s_1}{\boxed{M_1}} \vdash b \mid_n a_n \ \cdots \mid_1 a_1 \qquad \overset{s_2}{\boxed{M_2}} \vdash c\backslash b \mid_n a_n \ \cdots \mid_1 a_1 \cdots \mid_1 a_1}{\overset{s_1\ s_2}{\boxed{\begin{array}{l}\mathsf{do}\ m_1 \leftarrow M_1; m_2 \leftarrow M_2\\ \quad\boxed{m_1 \star \lambda v.m_2 \star \lambda k.v^\eta \star \lambda u_1.k^\eta \star \lambda f_1.(\lambda x_1.u_1(x_1) \star \cdots \star \lambda f_n.(\lambda x_n.u_n(x_n) \star \lambda w.f_n(x_n) \star \lambda g.w \star g)^\eta \cdots)^\eta}\end{array}}} \vdash c \mid_n a_n \ \cdots \mid_1 a_1}\ {<}\mathbf{S}_n$$

Figure 7: Probabilistic monadic CCG rule schemata.

3.2.5 Lexical knowledge

The introduction of the parameterized State.Probability monad allows us to model different forms of uncertainty as arising from the probabilistic knowledge we have about particular lexical items.

Consider again the sentence in (2).

(2) Jo ran a race.

We may analyze *Jo* as having the lexical entry in (17), where j is a constant of type *e*.

(17)
$$\overset{\textit{Jo}}{\boxed{\begin{array}{l}\texttt{do}\ \texttt{background}_{\textit{Jo}}\\ \quad \boxed{\mathsf{j}^{\eta^{\triangleright}}}\end{array}}} \vdash np$$

A couple of important pieces of notation are used here. First, $\texttt{background}_{\textit{Jo}}$ is a program of type $\mathbb{P}^{\sigma}_{\sigma'}\diamond$ which encodes our background knowledge about the lexical item *Jo*. This knowledge may be quite wide ranging. For instance, it might update an aspect of the state which encodes a phonetic representation of the utterances made in a discourse so far; or, it might add information to the common ground useful for later anaphora, i.e., that Jo is animate.

To flesh these ideas out, it will be useful to have operations that allow us to (probabilisticaly) read and write the values of stateful parameters. Given a state which is a tuple with a j^{th} component, we may wish to view this component in the course of defining a program.

Definition 3.23 ($\texttt{view}_{\pi_j}$).

$$\begin{aligned}\texttt{view}_{\pi_{|u|+1}} &: \mathbb{P}^{u\,\alpha\,v}_{u\,\alpha\,v}\alpha\\ \texttt{view}_{\pi_{|u|+1}} &= \lambda s.\boxed{\langle \pi_{|u|+1}(s), s\rangle}\end{aligned}$$

For example, if we wish to view the current common ground, we should bind this construct to a variable.

$$\begin{array}{l}\texttt{do}\cdots\\ \quad cg \leftarrow \texttt{view}_{\mathsf{CG}}\\ \quad\cdots\end{array}$$

As noted, $\mathsf{CG}(s)$ is the projection of the state *s* corresponding to its common ground parameter (whichever projection that may be).

In turn, we may wish to set a *new* common ground which modifies the old one in some way.

Definition 3.24 ($\texttt{set}_{\pi_j}$).

$$\begin{aligned}\texttt{set}_{\pi_{|u|+1}} &: \beta \to \mathbb{P}^{u\,\alpha\,v}_{u\,\beta\,v}\diamond\\ \texttt{set}_{\pi_{|u|+1}}(b) &= \lambda\langle\ldots,\_,\ldots\rangle.\boxed{\langle\diamond,\langle\ldots,b,\ldots\rangle\rangle}\end{aligned}$$

Thus a program which updates the current common ground using some function $f : \mathsf{P}\iota \to \mathsf{P}\iota$ can be defined by having it set the common ground after viewing it.

$$\begin{array}{l} \mathsf{do}\cdots \\ \quad cg \leftarrow \mathsf{view}_{\mathrm{CG}} \\ \quad \mathsf{set}_{\mathrm{CG}}(f(cg)) \\ \quad \cdots \end{array}$$

For example, if $\mathsf{background}_{\mathcal{J}o}$ updates the common ground to record that $\mathcal{J}o$ is animate, then it has the form in (18).[15]

(18) $$\mathsf{background}_{\mathcal{J}o} = \begin{array}{l} \mathsf{do}\cdots \\ \quad cg \leftarrow \mathsf{view}_{\mathrm{CG}} \\ \quad \mathsf{set}_{\mathrm{CG}}\begin{pmatrix} i \sim cg \\ \mathsf{observe}(\mathsf{animate}(w_i)(\mathsf{j})) \\ \boxed{i} \end{pmatrix} \\ \quad \cdots \end{array}$$

The second piece of new notation used in (17) is the operator $\triangleright$.

Definition 3.25 ($\triangleright$).

$$\begin{array}{rcl} (\cdot)^{\triangleright} & : & \mathsf{M}e \to \mathsf{M}e \\ m^{\triangleright} & = & m \star \lambda x, i, g.\{\langle x, x{::}g\rangle\} \end{array}$$

This operator takes a dynamic semantic value of type $\mathsf{M}e$ and gives back a value of the same type, but which has appended the (non-deterministic) entity it returns to the outgoing list of DRs.[16] Thus in the end, (17) encodes relevant probabilistic background knowledge and then returns a dynamic semantic value of type $\mathsf{M}e$ (consistent with its syntactic category), which adds j to the running list of DRs.

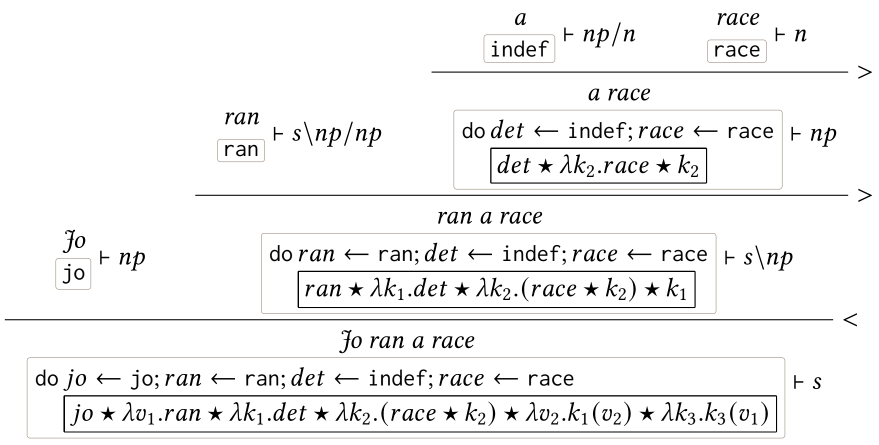
The indefinite determiner and the noun *race* can be treated similarly—in the former case, by adapting its lexical entry in (10) to the one in (19).

(19) $$\overset{a}{\boxed{\begin{array}{l} \mathsf{do}\;\mathsf{background}_{indef.} \\ \quad \boxed{(\lambda p, i, g.\{\langle x, x{::}g\rangle \mid p(x)(i)\})^{\eta}} \end{array}}} \vdash np/n$$

(20) $$\overset{race}{\boxed{\begin{array}{l} \mathsf{do}\;\mathsf{background}_{race} \\ \quad \boxed{\mathsf{race}^{\eta}} \end{array}}} \vdash n$$

[15] In order to save space, we will henceforth write '$\mathsf{background}$' once and concatenate the strings that name each program separately, in case we sequence more than one "background" program in a row; i.e., '$\mathsf{background}_{s_1 s_2}$' is shorthand for $\mathsf{background}_{s_1}; \mathsf{background}_{s_2}$.

[16] Both the notation and definition are adopted directly from Charlow 2014, §2.5.2, who encodes dynamic non-determinism via a State.Set monad; the difference here is that we also have an index to pass along.



a ⊢ np/n indef
race ⊢ n race
a race
do det ← indef; race ← race ⊢ np
det ⋆ λk2.race ⋆ k2
ran ⊢ s\np/np ran
ran a race
do ran ← ran; det ← indef; race ← race ⊢ s\np
ran ⋆ λk1.det ⋆ λk2.(race ⋆ k2) ⋆ k1
Jo ⊢ np jo
Jo ran a race
do jo ← jo; ran ← ran; det ← indef; race ← race ⊢ s
jo ⋆ λv1.ran ⋆ λk1.det ⋆ λk2.(race ⋆ k2) ⋆ λv2.k1(v2) ⋆ λk3.k3(v1)

Figure 8: Deriving (2).

Meanwhile, the lexical entry for *ran* may now be analyzed as introducing an *ambiguity*; e.g., between its locomotion sense and its management sense.

(21) $$\overset{\textit{ran}}{\boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{ran}\\ \quad run \leftarrow \mathsf{view}_{\mathsf{run}}\\ \quad \boxed{(\lambda x.(\lambda y.run(x)(y)^{\eta})^{\eta})^{\eta}}\end{array}}} \vdash s\backslash np/np$$

Given a state s, $\mathsf{view}_{\mathsf{run}}$ returns its projection $\mathsf{run}(s) : e \rightarrow e \rightarrow \iota \rightarrow t$. Since we are now trafficking in probability distributions over such states, we have an account of this ambiguity—both its probabilistic nature and its dependence on the state of the discourse.

Following our rule schemata, we can derive (2), in order to obtain a semantic value of type $\mathbb{P}^{\sigma}_{\sigma'}(\mathtt{M}(\iota \rightarrow t))$ (see Figure 8). To save space, we use the abbreviations 'jo', 'ran', etc., for the probabilistic semantic values provided above. Note that if we unpack these, we obtain the following result.

(22) $$\overset{\textit{Jo ran a race}}{\boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{\textit{Jo ran}}\\ \quad run \leftarrow \mathsf{view}_{\mathsf{run}}\\ \quad \mathsf{background}_{\textit{indef. race}}\\ \quad \boxed{\lambda i, g.\{\langle run(x)(\mathrm{j}), x{::}\mathrm{j}{::}g\rangle \mid \mathsf{race}(x)(i)\}}\end{array}}} \vdash s$$

These sorts of lexical entries exemplify our strategy for representing lexical knowledge in PDS. It is important to note that not all lexical knowledge, as we define it here, need pertain to linguistic properties per se; lexical background knowledge may contain whatever information about a lexical item might be relevant to the trajectory of an interpretation as it unfolds. Further, lexical knowledge may modify any aspect of the discourse state. We will demonstrate cases where it may modify the common ground, but it could just as well modify the QUD, as might be necessary under certain approaches to projective inferences (Simons 2007; Simons, Tonhauser, et al. 2010; Simons, Beaver, et al. 2017; Tonhauser, Beaver, and Degen 2018, i.a.).

3.2.6 Building discourses

Having illustrated our basic interpretation scheme, we turn to the question of how to integrate sentence interpretations into full discourses. Here, we define illocutionary operators that map expressions' semantic values onto discourses consisting of various utterance acts. We consider two such operators: `assert` and `ask`.

Making assertions. Given a sentence whose semantic value is $m : \mathbb{P}^{\sigma}_{\sigma'}(\mathtt{M}(\iota \rightarrow t))$, `assert` maps m to a discourse of type $\mathbb{P}^{\sigma}_{\sigma'}\diamond$.

Definition 3.26 (`assert`)**.**

$$\begin{aligned}
\mathtt{assert} \quad &: \quad \mathbb{P}^{\sigma}_{\sigma'}(\mathtt{M}(\iota \rightarrow t)) \rightarrow \mathbb{P}^{\sigma}_{\sigma'}\diamond \\
\mathtt{assert}(m) \quad &= \quad \mathtt{do}\, \phi \leftarrow m \\
&\qquad\quad drs \leftarrow \mathtt{view}_{\mathsf{DRs}} \\
&\qquad\quad cg \leftarrow \mathtt{view}_{\mathsf{CG}} \\
&\qquad\quad \mathtt{set}_{\mathsf{CG}}\begin{pmatrix} i \sim cg \\ \mathtt{observe}((drs \gg \phi)^{\exists,\epsilon}(i)) \\ \boxed{i} \end{pmatrix} \\
&\qquad\quad \mathtt{set}_{\mathsf{DRs}}(\lambda i, g.\{\langle \diamond, g'' \rangle \mid \exists g', p. \langle \diamond, g' \rangle \in drs(i)(g) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \wedge \langle p, g'' \rangle \in \phi(i)(g') \\
&\qquad\qquad\qquad\qquad\qquad\qquad \wedge p(i)\})
\end{aligned}$$

This definition has a few notable components and features some new notation, which we spell out. First, note the update to the projection of the discourse state, DRs. Given a state s, $\mathsf{DRs}(s) : \mathtt{M}\diamond$ is an index-sensitive relation on lists of DRs which we use to encode the current discourse's live anaphoric possibilities, including, e.g., those introduced by indefinite noun phrases. After retrieving this relation, an assertion of the dynamic proposition $\phi : \mathtt{M}(\iota \rightarrow t)$ *modifies* it by sequencing it with the dynamic updates introduced by ϕ, also taking into account the constraints imposed by the propositions it returns. Thus Definition 3.26 updates DRs by sequencing them with the monadic effects of the relevant semantic value, meanwhile continuing to return $\diamond$.

Second, the common ground of the current discourse state is updated so that the proposition $(drs \gg \phi)^{\exists} : \iota \rightarrow t$ is observed to hold of its indices. The operator $\gg$ here is defined in Definition 3.27.

Definition 3.27 ($\gg$)**.**

$$\begin{aligned}
(\gg) \quad &: \quad \mathtt{M}\alpha \rightarrow \mathtt{M}\beta \rightarrow \mathtt{M}\beta \\
m \gg n \quad &= \quad m \star \lambda\_.n
\end{aligned}$$

Given two monadic values, $\gg$ sequences their effects while keeping the returned value of only the second. Additionally, an existential closure operation is invoked, which we define in its general form.

Definition 3.28 (Generalized existential closure)**.**

$$\begin{aligned}
(\cdot_1)^{\exists,\cdot_2} \quad &: \quad \mathtt{M}(\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \iota \rightarrow t) \rightarrow \gamma \rightarrow \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \iota \rightarrow t \\
m^{\exists,g} \quad &= \quad \lambda x_1, \cdots, x_n, i.\exists\langle p, g' \rangle \in m(i)(g) : p(x_1)\cdots(x_n)(i)
\end{aligned}$$

$$\dfrac{\begin{array}{c}\textit{someone}\\ \boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{s.o.}\\ \quad\boxed{\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{animate}(w_i)(x)\}}\end{array}}\end{array}\vdash np \qquad \dfrac{\begin{array}{c}\textit{ran}\\ \boxed{\mathtt{ran}}\end{array}\vdash s\backslash np/np \qquad \dfrac{\begin{array}{c}\textit{a}\\ \boxed{\mathtt{indef}}\end{array}\vdash np/n \qquad \begin{array}{c}\textit{race}\\ \boxed{\mathtt{race}}\end{array}\vdash n}{\begin{array}{c}\textit{a race}\\ \boxed{\begin{array}{l}\mathsf{do}\ det \longleftarrow \mathtt{indef}; race \longleftarrow \mathtt{race}\\ \quad\boxed{det \star \lambda k_2.race \star k_2}\end{array}}\end{array}\vdash np}>}{\begin{array}{c}\textit{ran a race}\\ \boxed{\begin{array}{l}\mathsf{do}\ ran \longleftarrow \mathtt{ran}; det \longleftarrow \mathtt{indef}; race \longleftarrow \mathtt{race}\\ \quad\boxed{ran \star \lambda k_1.det \star \lambda k_2.(race \star k_2) \star k_1}\end{array}}\end{array}\vdash s\backslash np}>}{\begin{array}{c}\textit{someone ran a race}\\ \boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{s.o.}; ran \longleftarrow \mathtt{ran}; \mathsf{background}_{indef.}; \mathsf{background}_{race}\\ \quad\boxed{\lambda i,g.\{\langle ran \star \lambda k_1.k_1(x) \star \lambda k_2.k_2(y), x{::}y{::}g\rangle \mid \mathsf{animate}(w_i)(y) \wedge \mathsf{race}(w_i)(x)\}}\end{array}}\end{array}\vdash s}<$$

Figure 9: Deriving (24).

Note that ϵ—the empty list of DRs—is featured in the use of this operator in Definition 3.26. Thus fixed to the case of a dynamic proposition of the relevant form, generalized existential closure acts as in (23).

(23) $(drs \gg \phi)^{\exists,\epsilon} = \lambda i.\exists p, g, g'.\langle \diamond, g\rangle \in drs(i)(\epsilon) \wedge \langle p, g'\rangle \in \phi(i)(g) \wedge p(i)$

That is, we feed to the dynamic proposition ϕ the current state's index-sensitive relation on lists of DRs, applying it to the empty list ϵ, and then abstract over the index argument of the first projection of the members of the resulting set of proposition-list pairs, before existentially quantifying this set (while effectively constraining the relevant propositional witness so that it gives back T).

For illustration, consider the sentence in (24), which modifies (2) by adding yet another indefinite.

(24) Someone ran a race.

To *someone*, we assign the lexical entry in (25), which reflects the entry for the indefinite determiner in (19) (modulo the difference in type and the additional constraint that the relevant entities be animate).

(25) $$\begin{array}{c}\textit{someone}\\ \boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{s.o.}\\ \quad\boxed{\lambda i,g.\{\langle x, x{::}g\rangle \mid \mathsf{animate}(w_i)(x)\}}\end{array}}\end{array}\vdash np$$

A full derivation for (24) is given in Figure 9 (we again use mnemonic abbreviations for the full probabilistic semantic values). Note that this derivation departs minimally from the one given in Figure 8, the difference being the presence of the indefinite subject noun phrase.

We can follow Definition 3.26 in order to render the minimal discourse consisting of an assertion of this sentence's content in (26).

(26)
```
do $\mathtt{background}_{\mathit{s.o.\ ran}}$
    $\mathit{run} \leftarrow \mathtt{view}_{\mathsf{run}}$
    $\mathtt{background}_{\mathit{indef.\ race}}$
    $\mathit{drs} \leftarrow \mathtt{view}_{\mathsf{DRs}}$
    $\mathit{cg} \leftarrow \mathtt{view}_{\mathsf{CG}}$
    $\mathtt{set}_{\mathsf{CG}}\begin{pmatrix} i \sim \mathit{cg} \\ \mathtt{observe}(\exists y, x : \mathsf{animate}(w_i)(y) \wedge \mathsf{race}(w_i)(x) \wedge \mathit{run}(x)(y)(i)) \\ \boxed{i} \end{pmatrix}$
    $\mathtt{set}_{\mathsf{DRs}}(\lambda i, g.\{\langle \diamond, x{::}y{::}g' \rangle \mid \langle \diamond, g' \rangle \in \mathit{drs}(i)(g)$
                                        $\wedge\, \mathsf{animate}(w_i)(y)$
                                        $\wedge\, \mathsf{race}(w_i)(x)$
                                        $\wedge\, \mathit{run}(x)(y)(i)\})$
```

In sum, this discourse—after applying background information related to *someone* and *ran*—reads the semantic value of the latter from the current discourse state, whether that be, e.g., a management or a locomotion sense of the verb. Following this, it updates the common ground in order to filter out indices at which no person runs a race, otherwise keeping the probability distribution it encodes intact before introducing to the running index-sensitive relation on DRs the non-deterministic values x and y—a person and a race at the relevant index, respectively.

Note that the result of sequencing the dynamic proposition denoted by (24) with *drs* is negligble, in that the value to which this result is evaluated as the argument of the observe statement is not importantly affected. This holds because the semantic value of (24), once it is existentially closed, does not depend on the incoming list. If we instead considered the semantic value of a sentence with *pronouns* (e.g., *he ran a race*), the situation should turn out differently; i.e., the DRs of the discourse state which is updated would matter for how anaphora are resolved.

Asking questions. To model asking a question, we follow a categorial tradition that analyzes questions as denoting (what amount to) sets of true short answer meanings (Hausser and Zaefferer 1978; Hausser 1983; Xiang 2021; cf. Hamblin 1973; Karttunen 1977; Groenendijk and Stokhof 1984). Questions are thus of type $\alpha \rightarrow \iota \rightarrow t$, for some type α of short answers. Recall that in addition to *np*, *n*, and *s*, we have among our atomic categories ones for questions which represent what type of short answer they require; e.g., $q_{ind.}$ for individual questions, and $q_{deg.}$ for degree questions. The semantic types we assign to these categories reflect these semantic distinctions.

(27) $[\![q_{ind.}]\!]_{\mathsf{M}} = e \rightarrow \iota \rightarrow t$
$[\![q_{deg.}]\!]_{\mathsf{M}} = r \rightarrow \iota \rightarrow t$

Thus degree questions require short answers that have degree (i.e., real number) answers, while individual questions require entity answers.

Given an expression whose semantic value is a probabilistic dynamic question

$$m : \mathbb{P}^{\sigma}_{u\,\delta\,v}(\mathsf{M}(\alpha \rightarrow \iota \rightarrow t))$$

where δ is the type of the (polymorphic) QUD stack, the function ask maps m onto a discourse of type $\mathbb{P}^{\sigma}_{u\,((\alpha\rightarrow\iota\rightarrow t)\times\delta)\,v}\diamond$, i.e., which has the effect of pushing the question onto the stack (Roberts 2012; Farkas and Bruce 2010).

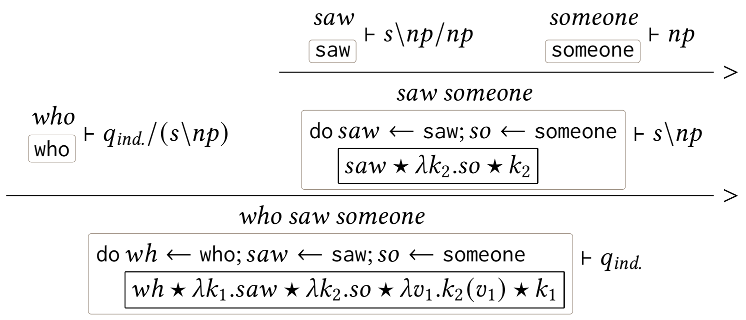

saw ⊢ s\np/np
someone ⊢ np
saw someone
do saw ← saw; so ← someone
saw ⋆ λk₂.so ⋆ k₂
⊢ s\np
who ⊢ q_ind./(s\np)
who saw someone
do wh ← who; saw ← saw; so ← someone
wh ⋆ λk₁.saw ⋆ λk₂.so ⋆ λv₁.k₂(v₁) ⋆ k₁
⊢ q_ind.

Figure 10: Deriving (28).

Definition 3.29 (ask)**.**

$$\begin{aligned}
\mathtt{ask} \quad &: \quad \mathbb{P}^{\sigma}_{u\,\delta\,v}(\mathtt{M}(\alpha \to \iota \to t)) \to \mathbb{P}^{\sigma}_{u\,((\alpha\to\iota\to t)\times\delta)\,v}{}^{\diamond} \\
\mathtt{ask}(m) \quad &= \quad \mathsf{do}\, q \leftarrow m \\
&\qquad\; drs \leftarrow \mathtt{view}_{\mathtt{DRs}} \\
&\qquad\; qs \leftarrow \mathtt{view}_{\mathtt{QUD}} \\
&\qquad\; \mathtt{set}_{\mathtt{QUD}}(\langle (drs \gg q)^{\exists,\epsilon}, qs\rangle)
\end{aligned}$$

Here, $qs : \delta$ (where δ is some n-ary product of question types) is the QUD stack associated with the current state. The effect of asking a question is to push a new semantic value of type $\alpha \to \iota \to t$—fashioned out of the dynamic semantic value returned by m—onto the stack. This semantic value is obtained by sequencing the dynamic semantic value returned by m with the current index-sensitive relation on DRs and then existentially closing, analogous to what is done for assertions. We assume that in general, asking a question does not update the state with new DRs; thus `ask` and `assert` are not completely analogous.

For illustration, consider the question in (28).

(28) Who saw someone?

A derivation of this question is given in Figure 10, using the lexical entries for *who* and *saw* in (29) and (30), respectively.

(29)
$$\overset{\textit{who}}{\boxed{\begin{array}{l}\mathsf{do}\,\mathsf{background}_{who} \\ \quad \boxed{(\lambda k, \_, g.\{\langle \lambda x, i.\mathsf{animate}(w_i)(x) \wedge k(x)^{\exists, x::g}(i), g\rangle\})}\end{array}}} \vdash q_{ind.}/(s\backslash np)$$

(30)
$$\overset{\textit{saw}}{\boxed{\begin{array}{l}\mathsf{do}\,\mathsf{background}_{saw} \\ \quad \boxed{(\lambda x.(\lambda y.(\lambda i.\mathsf{see}(w_i)(x)(y))^{\eta})^{\eta})^{\eta}}\end{array}}} \vdash s\backslash np/np$$

Two things are worth noting about the interpretation of *who*. First, that it imposes an animacy constraint on the entities of which the question it forms may be predicated at a given index.

Second, that it has a certain (internal) dynamic effect, analogous to that of *everyone* (see Figure 4 of Section 3.1.3): it feeds to its scope a list of DRs which has been updated with whichever entity it is predicated of at a given index.

Unpacking the final interpretation gives back the result in (31).

(31) $\texttt{do}\ \texttt{background}_{\textit{who saw s.o.}}$
$\boxed{\lambda i, g.\{\langle \lambda y, i'.\mathsf{animate}(w_{i'})(y) \wedge \mathsf{see}(w_{i'})(x)(y), x{::}g\rangle \mid \mathsf{animate}(w_i)(x)\}}$

Finally, asking this question produces the discourse in (32).

(32) $\texttt{do}\ \texttt{background}_{\textit{who saw s.o.}}$
$qs \leftarrow \texttt{view}_{\texttt{QUD}}$
$\texttt{set}_{\texttt{QUD}}(\langle \lambda y, i.\exists x.\mathsf{animate}(w_i)(x) \wedge \mathsf{animate}(w_i)(y) \wedge \mathsf{see}(w_i)(x)(y), qs\rangle)$

Note that, because the starting state's list of DRs does not ultimately affect the interpretation of the question, the corresponding view statement is eliminated.

Responding to questions. Given an ongoing discourse, an interlocutor may respond to the QUD at the top of the stack based on their prior knowledge. We may assume that a given responder has some background knowledge $bg : \mathtt{P}\sigma$ over *starting* discourse states, which they use—in conjuction with the interim updates to the discourse—to derive a probability distribution over answers to the current QUD. Given this prior, bg, and an ongoing discourse m, we aim to produce an answer distribution, of type $\mathtt{P}\alpha$ (for a QUD of type $\alpha \rightarrow \iota \rightarrow t$), which has the shape in (33).

(33) $s \sim bg$
$\langle \diamond, s' \rangle \sim m(s)$
$i \sim \mathsf{CG}(s')$
$\boxed{\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i))}$

Here, $\pi_1(\mathsf{QUD}(s'))$ is the QUD at the top of s''s QUD stack. Meanwhile, the returned value

$$\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i))$$

takes the maximum value of which this QUD is true, given the index i. Thus answers to the QUD are fundamentally based on the common ground: to compute their probability distribution, an index is sampled from the common ground, and the QUD is evaluated at that index. We assume, in general, that the type α of short answers to the QUD is associated with a join-semilattice and that the property denoted by the QUD is cumulative—$q(x)(i) \wedge q(y)(i) \rightarrow q(x \vee y)(i)$—so that max is defined. One type of question whose answer we might wish to model, for example, is a degree question; e.g., *how likely is it that S* (see Section 4.1). If we model degrees of likelihood as real numbers, we obtain the usual maximum operator under the semi-lattice where $x \vee y = \mathsf{max}(\{x, y\})$.

3.2.7 Linking models

As discussed in Section 1, a core feature of PDS is the support it provides for formal evaluability—i.e. the ability to explicitly quantitatively compare models. This support is implemented by explicitly modeling the relationship between a discourse and a particular data collection instrument whose values correspond to answers to the QUD at the top of the discourse's QUD stack. This approach, in turn, allows us to derive explicit probabilistic models (i) that differ either in the assumptions they make about the underlying probabilistic semantics or in the way that semantics is related to the response instrument; (ii) whose parameters may be fit to data collected using that instrument; and (iii) whose relative fit can be compared using standard statistical model comparison metrics.

We assume that a given instrument may be modeled as a family f of distributions representing the *likelihood*, which is then fixed by a collection Φ of nuisance parameters unrelated to the discourse(s) of interest. Thus we may define a family of *response functions*, parametric in the particular data collection instrument—i.e., a likelihood function. Each such function takes a distribution representing one's background knowledge bg, along with an ongoing discourse m, to produce a distribution over answers to the current QUD, given the data collection instrument.

Definition 3.30 (Response function).

$$\begin{aligned}
\mathtt{respond}^{f_\Phi : \alpha \to \mathsf{P}\beta} &: \mathsf{P}\sigma \to \mathbb{P}^{\sigma}_{u\,(\alpha \to \iota \to t) \times \delta\, v}{}^{\diamond} \to \mathsf{P}\beta \\
\mathtt{respond}^{f_\Phi}(bg)(m) &= s \sim bg \\
&\quad\ \langle \diamond, s' \rangle \sim m(s) \\
&\quad\ i \sim \mathsf{CG}(s') \\
&\quad\ f(\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i)), \Phi)
\end{aligned}$$

For example, suppose we present an experimental participant with (34) and provided them with two radio buttons labeled '*Jo*' and '*Mo*', respectively—i.e., having the constraint that exactly one button must be selected before proceeding.

(34) a. Jo left.
     b. Mo stayed.
     c. Which person left?

Viewing (34) as a *discourse*, we can assign it an interpretation like (35) (where we abbreviate the probabilistic semantic values of individual sentences).

(35)
```
do assert(jo_left)
   assert(mo_stayed)
   ask(which_person_left)
```

Because the data collection instrument consists of two radio buttons, the response function should produce a probability distribution over strings of one bit—i.e., a Bernoulli distribution.

Definition 3.31 (Bernoulli distribution).

$$\mathtt{Bern} : r \to \mathsf{P}t$$

Given a real number $p \in [0,1]$, $\mathtt{Bern}(p)$ returns $\mathtt{T}$ with probability p and $\mathtt{F}$ with probability $1-p$. Thus a suitable definition of the response function would be the one in (36).

(36)

$$\begin{aligned}\mathtt{respond}^{\lambda x.\mathtt{Bern}(\mathbb{1}(x \in \{\mathsf{j}\}))}(bg)(m) \quad = \quad & s \sim bg \\ & \langle \diamond, s' \rangle \sim m(s) \\ & i \sim \mathsf{CG}(s') \\ & \mathtt{Bern}(\mathbb{1}(\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i)) \in \{\mathsf{j}\}))\end{aligned}$$

Here, the response distribution is Bernoulli with parameter either 1 or 0, depending on whether or not the maximal answer to the QUD is an element of the singleton set containing the (static) semantic value of *Jo*. Thus if this maximal answer is j, then the response is predicted to take a degenerate distribution all of whose mass is assigned to T—i.e., such that the '*Jo*' button is active; whereas if the maximal answer is something else (e.g., m), then the response is predicted to take a degenerate distribution all of whose mass is assigned to F—i.e., such that the other, '*Mo*' button is active.

For an example featuring somewhat more uncertainty, consider the discourse in (37).

(37) a. Jo, Mo, and Bo left.
b. Two of them returned.
c. Who returned?

Now imagine that the data collection instrument consists of three check boxes, labeled '*Jo*', '*Mo*', and '*Bo*'. Here, we relax the constraint that the choices are mutually exclusive and that one box need be selected in order to proceed. In this case, the response function should produce a probability distribution over strings of three bits; i.e., a trivariate Bernoulli distribution, which is defined as taking $2^3 - 1 = 7$ parameters.

Definition 3.32 (Trivariate Bernoulli distribution)**.**

$$\mathtt{TriBern} \;:\; r^7 \longrightarrow \mathsf{P}t^3$$

As with the (univariate) Bernoulli distribution, each parameter $p_i \in [0,1]$, but now with the additional constraint that $\sum_{i=1}^{7} p_i \leq 1$. Meanwhile, the resulting distribution is over a triple of truth values, each T or F as according to whether or not the box labeled '*Jo*', '*Mo*', or '*Bo*' is checked, respectively. Under these assumptions, a suitable definition of the response function would be the one in (38).

(38)

$$\begin{aligned}& \mathtt{respond}^{\lambda x.\mathtt{TriBern}(...)}(bg)(m) \\ = \;& s \sim bg \\ & \langle \diamond, s' \rangle \sim m(s) \\ & i \sim \mathsf{CG}(s') \\ & \mathtt{TriBern}(\mathbb{1}(\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i)) = \mathsf{j}), ..., \mathbb{1}(\mathsf{max}(\lambda x.\pi_1(\mathsf{QUD}(s'))(x)(i)) = \mathsf{j} \lor \mathsf{m} \lor \mathsf{b}))\end{aligned}$$

Thus the response distribution is again degenerate—all of the mass is assigned to one of the eight possible combinations of three truth values—but the exact nature of its degeneracy is now determined by seven parameters, each reflecting an element of the join-semilattice generated by Jo, Mo, and Bo.

Crucially, the discourse in (37)—as opposed to the one in (34)—is associated with significant uncertainty about the answer to its question, *Who returned?*. Unless one has strong priors about Jo, Mo, and Bo, any two-person answer seems about as good as any other. One might wonder if this uncertainty is actually modeled by the response distribution schematized in (38); after all, the likelihood itself is degenerate and thus not much of a probability distribution!

In fact, the full model represented by (38) may nonetheless encode a significant amount of uncertainty, resulting in any number of distributions over t^3. This uncertainty stems not from the trivariate Bernoulli likelihood, however, but from the index i which is sampled from the common ground: different indices will provide different maximal answers to the QUD; meanwhile, the total uncertainty will be reflected in the relative flatness of the distribution over these answers.

Thus besides providing dynamic semantic analyses of the linguistic expressions which may constitute some set of experimental materials—together with the structure of the relevant linking model—PDS recognizes the importance of the assumptions one makes about general background knowledge. All three of these kinds of assumptions are crucial, and they must be specified formally before one evaluates a PDS model to a statistical model that can be fit to data. We return to this point in Section 5, where we introduce the final piece of technology needed to perform this evaluation: the notion of a density function. Before taking this final step, however, we explore the explanatory capabilities of the technology we have already introduced a bit further in the next section—focusing in particular on its ability to capture fine-grained distinctions among different forms of uncertainty.

4 Implementations of the framework

As we discuss in Section 2, one of the main benefits of PDS is that it allows us to naturally distinguish different forms of uncertainty, which broadly fall into the categories of *metalinguistic uncertainty* and *occasional uncertainty*. This section aims to demonstrate how it captures this distinction. We discuss the formal mechanisms by which the distinction is captured in Section 4.1, turning to specific case studies in Sections 4.2 and 4.3.

We should note that while we take a particular position on which phenomena should be associated with which kinds of uncertainty in this paper, the matter is ultimately empirical. Models fit to linguistic inference data may turn out to perform best—as assessed by standard statistical model comparison metrics—when they regard some source of uncertainty in a way contrary to what we conjecture. Indeed, it is in this way that PDS provides a methodology for performing theory comparison—i.e., by implementing statistical models of linguistic datasets—in addition to a theoretical framework for probabilistic semantics.

4.1 Explaining sources of uncertainty

In general, the difference between metalinguistic and occasional uncertainty is reflected in the quality of the responses produced, given a particular question. Questions that interrogate degrees of likelihood, for example, will elicit *discrete* inferences about sources of uncertainty which are metalinguistic in nature, but *gradient* inferences about sources of uncertainty which are occasional in nature. Here is why.

Assume that the relevant question has the form in (39).

(39) How likely is it that S?

To assign a semantics to the modal adjective *likely*, we assume that modal adjective phrases have the same semantic type as sentences. Degree-denoting expressions, moreover, are interpreted as real numbers.

(40) $[\![ap_{mod.}]\!]_{\mathsf{M}} = \iota \rightarrow t$
$[\![d]\!]_{\mathsf{M}} = r$

Likely itself has the category $ap_{mod.}\backslash d/s$: it selects a sentence, along with a degree-denoting argument, in order to form an $ap_{mod.}$; thus its semantic type is $\mathsf{M}((\iota \rightarrow t) \rightarrow \mathsf{M}(r \rightarrow \mathsf{M}(\iota \rightarrow t)))$. The full lexical entry we assign it is in (41) (to be modified in (60) of Section 4.3).

(41)
$$\begin{array}{c} likely \\ \boxed{\begin{array}{l}\texttt{do}\ \texttt{background}_{likely} \\ \quad cg \leftarrow \texttt{view}_{\texttt{CG}} \\ \quad \boxed{(\lambda\phi.(\lambda d.(\lambda\_.Pr\begin{pmatrix} i \sim cg \\ \boxed{\phi(i)}\end{pmatrix} \geq d)^{\eta})^{\eta})^{\eta}}\end{array}} \end{array} \vdash ap_{mod.}\backslash d/s$$

According to the semantic part of this lexical entry, *likely* computes the probability that its prejacent is true, given the common ground; it then compares this probability to the abstracted degree threshold ('d'). As for probabilities, they can be computed from programs of type $\mathsf{P}t$ in terms of a more general expected value operator, following Definition 4.1.[17]

Definition 4.1 (Expected value operator)**.**

$$\begin{aligned} E_{(\cdot)} &: \mathsf{P}\alpha \rightarrow (\alpha \rightarrow r) \rightarrow r \\ Pr &: \mathsf{P}t \rightarrow r \\ Pr(m) &= E_m(\mathbb{1}) \end{aligned}$$

That is, a probability is just the expected value of the indicator function (a fact well-known from probability theory).

The *wh*-word *how* can be given a semantics according to which it feeds its adjectival complement back to the context from which it was extracted, while hanging on to the adjective's degree argument.

[17]Following our practice re: `factor`, we take $E_{(\cdot)}$ to be a primitive of the language of probabilistic programs (i.e., a constant). See Grove and Bernardy 2023 for an implementation of this operator using continuations.

$$\dfrac{\dfrac{\overset{\textit{how}}{\boxed{\texttt{how}}} \vdash q_{deg.}/(s/(ap_{mod.}/s))/(ap_{mod.}\backslash d/s) \qquad \overset{\textit{likely}}{\boxed{\texttt{likely}}} \vdash ap_{mod.}\backslash d/s}{\overset{\textit{how likely}}{\boxed{\begin{array}{l}\texttt{do}\ wh \leftarrow \texttt{how}; l \leftarrow \texttt{likely}\\ \quad \boxed{wh \star \lambda k_1.l \star k_1}\end{array}}} \vdash q_{deg.}/(s/(ap_{mod.}/s))}> \qquad \dfrac{\overset{\textit{is it that}}{\boxed{\texttt{is\_it\_that}}} \vdash s/(ap_{mod.}/s)/s \qquad \overset{S}{\boxed{\texttt{S}}} \vdash s}{\overset{\textit{is it that S}}{\boxed{\begin{array}{l}\texttt{do}\ iit \leftarrow \texttt{is\_it\_that}; \phi \leftarrow \texttt{S}\\ \quad \boxed{iit \star \lambda k_3.\phi \star k_3}\end{array}}} \vdash s/(ap_{mod.}/s)}>}{\overset{\textit{how likely is it that S}}{\boxed{\begin{array}{l}\texttt{do}\ wh \leftarrow \texttt{how}; l \leftarrow \texttt{likely}; iit \leftarrow \texttt{is\_it\_that}; \phi \leftarrow \texttt{S}\\ \quad \boxed{wh \star \lambda k_1.l \star \lambda v_1.k_1(v_1) \star \lambda k_2.iit \star \lambda k_3.\phi \star \lambda v_2.k_3(v_2) \star k_2}\end{array}}} \vdash q_{deg.}}>$$

Figure 11: Deriving (39).

(42) $$\overset{\textit{how}}{\boxed{\begin{array}{l}\texttt{do background}_{how}\\ \quad \boxed{(\lambda f.(\lambda \phi, \_, g.\{\langle (\lambda d, i.\phi(\lambda p.f(p) \star \lambda v.v(d))^{\exists,g}(i), g\rangle\})^{\eta})^{\eta}}\end{array}}} \vdash q_{deg.}/(s/(ap_{mod.}/s))/(ap_{mod.}\backslash d/s)$$

For current purposes, we regard the string *is it that* as a constituent and assign it a meaning that merely chases functions around.

(43) $$\overset{\textit{is it that}}{\boxed{\begin{array}{l}\texttt{do background}_{is\ it\ that}\\ \quad \boxed{(\lambda \phi.(\lambda f.f(\phi))^{\eta})^{\eta}}\end{array}}} \vdash s/(s/ap_{mod.})/s$$

A derivation of (39) is provided in Figure 11. Reducing its last term yields the result in (44).

(44) $$\begin{array}{l}\texttt{do background}_{how\ likely}\\ \quad cg \leftarrow \texttt{view}_{\texttt{CG}}\\ \quad \texttt{background}_{is\ it\ that}\\ \quad \phi \leftarrow \texttt{S}\\ \quad \boxed{\phi \star \lambda p.(\lambda d, \_.Pr\begin{pmatrix} i \sim cg\\ \boxed{p(i)}\end{pmatrix} \geq d)^{\eta}}\end{array}$$

Asking (39) therefore produces the discourse in (45).

(45) $$\begin{array}{l}\texttt{do background}_{how\ likely}\\ \quad cg \leftarrow \texttt{view}_{\texttt{CG}}\\ \quad \texttt{background}_{is\ it\ that}\\ \quad \phi \leftarrow \texttt{S}\\ \quad drs \leftarrow \texttt{view}_{\texttt{DRs}}\\ \quad qs \leftarrow \texttt{view}_{\texttt{QUD}}\\ \quad \texttt{set}_{\texttt{QUD}}(\langle \lambda d.(drs \gg \phi \star \lambda p.(Pr\begin{pmatrix} i \sim cg\\ \boxed{p(i)}\end{pmatrix} \geq d)^{\eta})^{\exists,\epsilon}, qs\rangle)\end{array}$$

It will become apparent when considering particular examples of prejacents S that the maximum degree d such that

$$Pr\left(\begin{array}{c} i \sim cg \\ \boxed{p(i)} \end{array}\right) \geq d$$

will be either 0 or 1 when S exhibits only metalinguistic uncertainty, while it will potentially be any value in $[0, 1]$ when S exhibits occasional uncertainty (possibly, in addition to metalinguistic uncertainty). Schematically, this is because if p's value does not in fact depend on i (i.e., if it exhibits no occasional uncertainty), its probability in the common ground will be either 0 or 1; if p's value does depend on i, this probability will potentially be middling. This point should become clearer as we consider specific examples.

In Sections 4.2 and 4.3, we provide examples of metalinguistic and occasional uncertainty, respectively. In Section 4.2, we focus on anaphora resolution, and in Section 4.3, we treat vagueness. Granted, these two topics provide only a sample of the phenomena that could potentially fall into either category—however, we believe they give an adequate gist of the framework and how it may be deployed in practice.

4.2 Metalinguistic uncertainty

One well studied source of metalinguistic uncertainty is the anaphora resolution choices made in particular discourses. Consider the mini-discourse in (46), resembling (1).

(46) a. A magician scowled and their assistant smirked.

b. They were pleased.

There is again an ambiguity about the antecedent of the pronoun; e.g., it may refer to the magician, the magician's assistant, or the individual sum of them both. For simplicity, we only treat the singular interpretation of *they* here. Thus to countenance this metalinguistic uncertainty, we can assign the pronoun the lexical entry in (47).

(47)

$$\begin{array}{c} \textit{they} \\ \boxed{\begin{array}{l} \texttt{do}\ \texttt{background}_{they} \\ \quad cg \leftarrow \texttt{view}_{\texttt{CG}} \\ \quad i \leftarrow cg^{\uparrow} \\ \quad sel \leftarrow \texttt{view}_{\texttt{sel}} \\ \quad \texttt{observe}(\forall g.\texttt{animate}(w_i)(sel(g)))^{\uparrow} \\ \quad \boxed{\lambda\_, g.\{\langle sel(g), g\rangle\}} \end{array}} \end{array} \vdash np$$

The meaning associated with this lexical entry samples an index from the common ground and invokes an observe statement, both of which, moreover, have been lifted from a type in the monad P to a type in the parameterized monad $\mathbb{P}$.

Definition 4.2 (Lift)**.**

$$\begin{array}{rcl} (\cdot)^{\uparrow} & : & \mathsf{P}\alpha \longrightarrow \mathbb{P}^{\sigma}_{\sigma}\alpha \\ m^{\uparrow} & = & \lambda s.v \sim m \\ & & \quad \boxed{\langle v, s\rangle} \end{array}$$

Thus this meaning reads a selection function *sel* from the state—something of type $\gamma \rightarrow e$—and ensures that its range is inhabited by entities which are animate at every index of the common ground. The monadic value it returns reads in a list of DRs and uses *sel* to choose one.

An analogous lexical entry can be provided for the possessive pronoun *their*.

(48)
$$\begin{array}{c} \textit{their} \\ \boxed{\begin{array}{l} \texttt{do}\ \texttt{background}_{\textit{their}} \\ \quad cg \leftarrow \texttt{view}_{\texttt{CG}} \\ \quad i \leftarrow cg^{\uparrow} \\ \quad sel \leftarrow \texttt{view}_{\texttt{sel}} \\ \quad \texttt{observe}(\forall g.\texttt{animate}(w_i)(sel(g)))^{\uparrow} \\ \quad poss \leftarrow \texttt{view}_{\texttt{poss}} \\ \quad \boxed{(\lambda p, i, g.\{\langle x, x{::}g\rangle \mid p(x)(i) \wedge poss(x)(sel(g))(i)\})^{\eta}} \end{array}} \end{array} \vdash np/n$$

Here, we analyze the possessive pronoun as analogous to an indefinite determiner, such that the referent of the noun phrase in which it results is non-deterministic (and, moreover, need not be unique; see Coppock and Beaver 2015). Note, further, that the relation holding between this referent and the antecedent of the pronoun is itself uncertain: we assume that there is a metalinguistic parameter $\texttt{poss} : e \rightarrow e \rightarrow \iota \rightarrow t$ which relates the two entities at a given index.

We analyze the copula as in (49), so that it turns an adjective phrase into an expression taking a noun phrase in order to give back a sentence.

(49)
$$\begin{array}{c} \textit{were} \\ \boxed{\begin{array}{l} \texttt{do}\quad \texttt{background}_{\textit{were}} \\ \quad \boxed{(\lambda f.(\lambda x.f(x)^{\eta})^{\eta})^{\eta}} \end{array}} \end{array} \vdash s\backslash np/ap$$

Meanwhile, the coordinator *and* gets the lexical entry in (50).

(50)
$$\begin{array}{c} \textit{and} \\ \boxed{\begin{array}{l} \texttt{do}\ \texttt{background}_{\textit{and}} \\ \quad \boxed{(\lambda q.(\lambda p.(\lambda i.p(i) \wedge q(i))^{\eta})^{\eta})^{\eta}} \end{array}} \end{array} \vdash s\backslash s/s$$

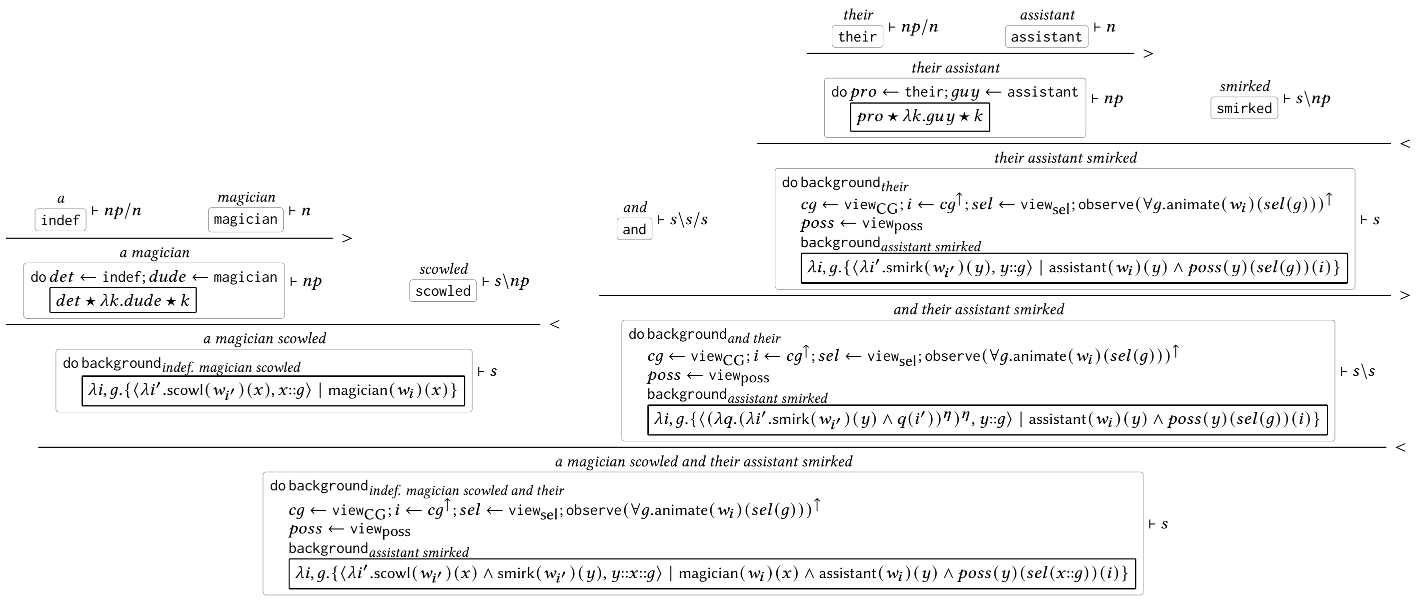
Assuming the obvious lexical entries for the other expressions, we provide the derivation of (46a) in Figure 12 and the derivation of (46b) in Figure 13.

Imagine now that instead of (46b), we followed (46a) by the question in (51b).

(51) a. A magician scowled and their assistant smirked.

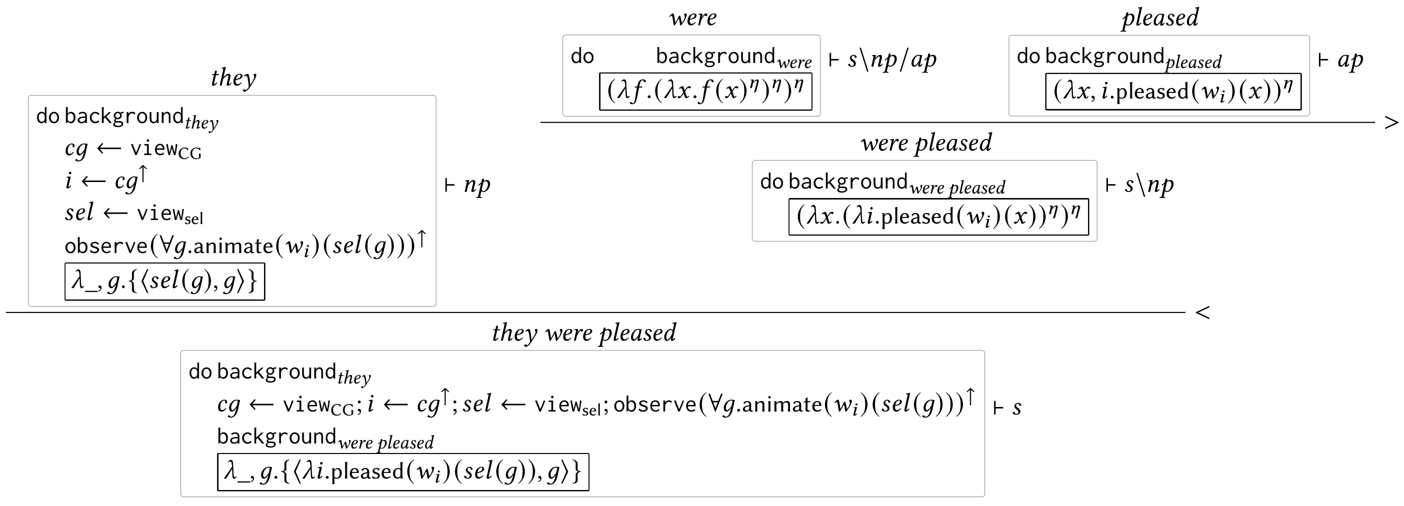
b. How likely is it that they were pleased?

By adopting the template derived in (45) of Section 4.1 and substituting the interpretation that results in Figure 13, we can derive (in (52)) the discourse that consists of asking (51b).



their ⊢ np/n
assistant ⊢ n
their assistant
do pro ← their; guy ← assistant
pro ⋆ λk.guy ⋆ k ⊢ np
smirked ⊢ s\np
their assistant smirked
do background_their
cg ← view_CG; i ← cg↑; sel ← view_sel; observe(∀g.animate(w_i)(sel(g)))↑
poss ← view_poss
background_assistant smirked
λi, g.{⟨λi′.smirk(w_i′)(y), y::g⟩ | assistant(w_i)(y) ∧ poss(y)(sel(g))(i)} ⊢ s
and ⊢ s\s/s
and their assistant smirked
do background_and their
cg ← view_CG; i ← cg↑; sel ← view_sel; observe(∀g.animate(w_i)(sel(g)))↑
poss ← view_poss
background_assistant smirked
λi, g.{⟨(λq.(λi′.smirk(w_i′)(y) ∧ q(i′))^η)^η, y::g⟩ | assistant(w_i)(y) ∧ poss(y)(sel(g))(i)} ⊢ s\s
a ⊢ np/n
magician ⊢ n
a magician
do det ← indef; dude ← magician
det ⋆ λk.dude ⋆ k ⊢ np
scowled ⊢ s\np
a magician scowled
do background_indef. magician scowled
λi, g.{⟨(λi′.scowl(w_i′)(x), x::g⟩ | magician(w_i)(x)} ⊢ s
a magician scowled and their assistant smirked
do background_indef. magician scowled and their
cg ← view_CG; i ← cg↑; sel ← view_sel; observe(∀g.animate(w_i)(sel(g)))↑
poss ← view_poss
background_assistant smirked
λi, g.{⟨λi′.scowl(w_i′)(x) ∧ smirk(w_i′)(y), y::x::g⟩ | magician(w_i)(x) ∧ assistant(w_i)(y) ∧ poss(y)(sel(x::g))(i)} ⊢ s

Figure 12: Deriving (46a).



were
do background_were
(λf.(λx.f(x)^η)^η)^η ⊢ s\np/ap
pleased
do background_pleased
(λx, i.pleased(w_i)(x))^η ⊢ ap
were pleased
do background_were pleased
(λx.(λi.pleased(w_i)(x))^η)^η ⊢ s\np
they
do background_they
cg ← view_CG
i ← cg↑
sel ← view_sel
observe(∀g.animate(w_i)(sel(g)))↑
λ_, g.{⟨sel(g), g⟩} ⊢ np
they were pleased
do background_they
cg ← view_CG; i ← cg↑; sel ← view_sel; observe(∀g.animate(w_i)(sel(g)))↑
background_were pleased
λ_, g.{⟨λi.pleased(w_i)(sel(g)), g⟩} ⊢ s

Figure 13: Deriving (46b).

(52)
```
do $\mathtt{background}_{\textit{how likely is it that they}}$
   $cg \leftarrow \mathtt{view}_{\mathrm{CG}}$
   $i \leftarrow cg^{\upharpoonleft}$
   $sel \leftarrow \mathtt{view}_{\mathrm{sel}}$
   $\mathtt{observe}(\forall g.\mathrm{animate}(w_i)(sel(g)))^{\upharpoonleft}$
   $\mathtt{background}_{\textit{were pleased}}$
   $drs \leftarrow \mathtt{view}_{\mathrm{DRs}}$
   $qs \leftarrow \mathtt{view}_{\mathrm{QUD}}$
   $\mathtt{set}_{\mathrm{QUD}}(\langle \lambda d.(drs \gg (\lambda\_, g.\{\langle Pr\left(\begin{array}{l} i \sim cg \\ \boxed{\mathrm{pleased}(w_i)(sel(g))} \end{array}\right) \geq d, g\rangle\}))^{\exists,\epsilon}, qs\rangle)$
```

To go a step further, we may now assign a meaning to the entirety of (51) by sequencing the result of asserting (46a) with (52).

(53)
$$
\begin{array}{l}
\mathsf{do}\ \mathtt{background}_{\textit{indef. magician scowled and their}}\\
\quad cg \leftarrow \mathtt{view}_{\mathrm{CG}}; i \leftarrow cg^{\uparrow}; sel \leftarrow \mathtt{view}_{\mathrm{sel}}; \mathtt{observe}(\forall g.\mathsf{animate}(w_i)(sel(g)))^{\uparrow}\\
\quad poss \leftarrow \mathtt{view}_{\mathrm{poss}}\\
\quad \mathtt{background}_{\textit{assistant smirked}}\\
\quad drs \leftarrow \mathtt{view}_{\mathrm{DRs}}\\
\quad cg' \leftarrow \mathtt{view}_{\mathrm{CG}}\\
\quad \mathtt{set}_{\mathrm{CG}}\left(\begin{array}{l} i' \sim cg'\\ \mathtt{observe}\left(\begin{array}{l}\exists g, x, y.\ \langle \diamond, g\rangle \in drs(i')(\epsilon)\\ \qquad\qquad \wedge\ \mathsf{magician}(w_{i'})(x)\\ \qquad\qquad \wedge\ \mathsf{scowl}(w_{i'})(x)\\ \qquad\qquad \wedge\ \mathsf{assistant}(w_{i'})(y)\\ \qquad\qquad \wedge\ poss(y)(sel(x{::}g))(i')\\ \qquad\qquad \wedge\ \mathsf{smirk}(w_{i'})(y)\end{array}\right)\\ \boxed{i'}\end{array}\right)\\
\quad \mathtt{set}_{\mathrm{DRs}}(\lambda i, g.\{\langle \diamond, y{::}x{::}g'\rangle \mid \langle \diamond, g'\rangle \in drs(i)(g)\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{magician}(w_i)(x)\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{scowl}(w_i)(x)\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{assistant}(w_i)(y)\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ poss(y)(sel(x{::}g))(i)\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{smirk}(w_i)(y)\})\\
\quad \mathtt{background}_{\textit{how likely}}\\
\quad cg'' \leftarrow \mathtt{view}_{\mathrm{CG}}\\
\quad \mathtt{background}_{\textit{is it that they}}\\
\quad cg''' \leftarrow \mathtt{view}_{\mathrm{CG}}; i''' \leftarrow cg'''^{\uparrow}; sel' \leftarrow \mathtt{view}_{\mathrm{sel}}; \mathtt{observe}(\forall g.\mathsf{animate}(w_{i'''})(sel'(g)))^{\uparrow}\\
\quad qs \leftarrow \mathtt{view}_{\mathrm{QUD}}\\
\quad \mathtt{set}_{\mathrm{QUD}}(\langle \lambda d, i.\exists g, x, y.\ \langle \diamond, g\rangle \in drs(i)(\epsilon)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{magician}(w_i)(x)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{scowl}(w_i)(x)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{assistant}(w_i)(y)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ poss(y)(sel(x{::}g))(i)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{smirk}(w_i)(y)\\
\qquad\qquad\qquad\qquad\qquad \wedge\ Pr\left(\begin{array}{l} i'' \sim cg''\\ \boxed{\mathsf{pleased}(w_{i''})(sel'(y{::}x{::}g))}\end{array}\right) \geq d, qs\rangle)
\end{array}
$$

The final act represented in (53) is that of pushing a new question onto the QUD stack. When a response is made to this question, it will involve sampling an index from the common ground and taking the maximum degree of which the question is true at that index. Thus at each index, we obtain as a response the maximum degree such that there is a scowling magician along with their smirking assistant, and such that the probability that the antecedent of *they* was pleased in the common ground bounds that degree from above. Crucially, the antecedent of *they* at each index may in fact *be* either that index's scowling magician or its smirking assistant; as a result, we obtain the probability that whichever it is was pleased in the common ground. These appear to be more-or-less adequate truth conditions.

Meanwhile, the major lesson to observe here is that the probability that *they were pleased* is true at a given index depends on how the pronoun *they* is resolved: if, for example, it is resolved to the scowling magician ('x'), this probability may be low—say, if scowling and being pleased negatively covary; but if it is resolved to the smirking assistant ('y'), the probability may be higher—i.e., if smirking and being pleased positively covary. In the former case, the resulting distribution of probabilities may hover around, say, 0.1, while in the latter, it may hover around, say, 0.7. Crucially, the answer distribution for (53) will in this case be *bimodal*, with each mode corresponding to a way of resolving the antecedent of *they*. Such multimodality is a signature characteristic of metalinguistic uncertainty. That is, whereas occasional uncertainty may *interact* with linguistic operators such as *likely*—resulting in it being *integrated over*—metalinguistic uncertainty directly affects response distributions.

4.3 Occasional uncertainty

The uncertainty exhibited by vague inferences may appear difficult to classify due to the fact that forming *how likely* questions out of examples such as (3), as in (54), gives rise to answers which are made on the basis of world knowledge—here, about Jo's height.

(54) How likely is it that Jo is tall?

An answer to (54) might be *pretty likely*, for instance, based entirely on the uncertainty about whether or not Jo's height passes some fixed threshold.

Meanwhile, if it is assumed to be background knowledge that Jo is six feet tall, (54) becomes significantly less felicitous. In particular, (54) appears to imply ignorance about Jo's height. Thus in order to effectively probe uncertainty about the height threshold introduced by *tall*, it is necessary to ask a question more akin to (55b), formed from the prejacent in (55a).[18]

(55) a. Bo considers Jo tall.
     b. How likely is it that Bo considers Jo tall?

To approach these facts, we can analyze *considers* as manipulating the context part of the index fed to its prejacent in a way that depends on its subject. To do this, let us assume that there is a projection c of indices with the type in (56), where $\iota, \iota' \in \mathcal{S}_\iota$.

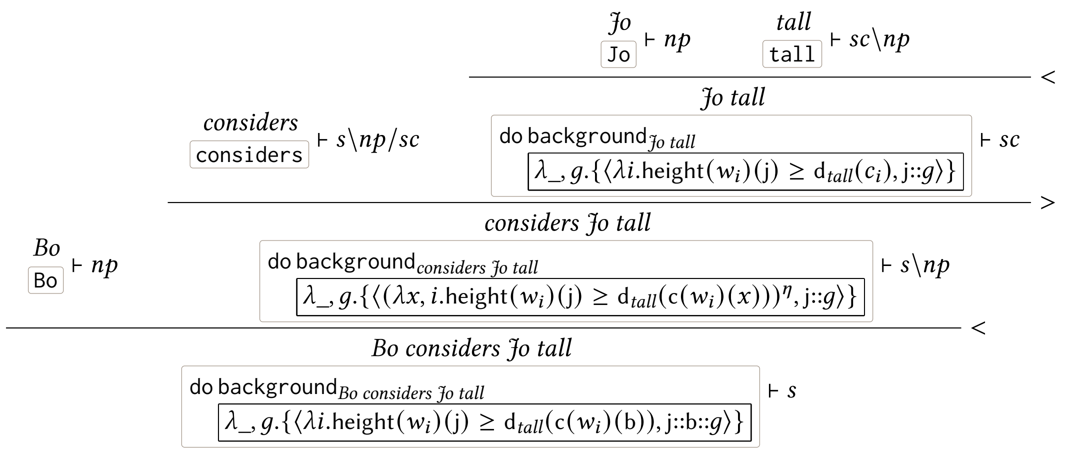
(56) $\mathsf{c} : \iota \to e \to \kappa_{\iota'}$

Further, because *considers* selects a small-clause argument, its syntactic type is $s\backslash np/sc$. Meanwhile, we associate small clauses with the same semantic type as sentences.

(57) $[\![sc]\!]_\mathsf{M} = \iota \to t$

We can then assign *considers* the lexical entry in (58).

[18]Or a variant thereof, e.g., *Jo is considered tall.*



Jo
Jo ⊢ np
tall
tall ⊢ sc\np
<
Jo tall
do background_{Jo tall} λ_, g.{⟨λi.height(w_i)(j) ≥ d_tall(c_i), j::g⟩} ⊢ sc
considers
considers ⊢ s\np/sc
>
considers Jo tall
do background_{considers Jo tall} λ_, g.{⟨(λx, i.height(w_i)(j) ≥ d_tall(c(w_i)(x)))^η, j::g⟩} ⊢ s\np
Bo
Bo ⊢ np
<
Bo considers Jo tall
do background_{Bo considers Jo tall} λ_, g.{⟨λi.height(w_i)(j) ≥ d_tall(c(w_i)(b)), j::b::g⟩} ⊢ s

Figure 14: Deriving (55a).

(58)
$$\overset{\textit{considers}}{\boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{\textit{considers}}\\ \quad\boxed{(\lambda\phi.(\lambda x.(\lambda i'.\phi(i_{w_{i'},\mathsf{c}(w_{i'})(x)}))^{\eta})^{\eta})^{\eta}}\end{array}}} \vdash s\backslash np/sc$$

Thus given an index i' and an entity x, *considers* feeds its prejacent the new index $i_{w_{i'},\mathsf{c}(w_{i'})(x)}$. Following Definition 3.19 of Section 3.2.2, this is the index formed from the world part of i' and the new context gotten by applying c to i' and x.

Finally, for current purposes, we assume that *tall* takes a noun phrase to produce a small clause, as in (59).

(59)
$$\overset{\textit{tall}}{\boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{\textit{tall}}\\ \quad\boxed{(\lambda x.(\lambda i.\mathsf{height}(w_i)(x) \geq \mathsf{d}_{\textit{tall}}(c_i))^{\eta})^{\eta}}\end{array}}} \vdash sc\backslash np$$

We may then derive (55a) as in Figure 14.

To get a handle on the infelicity of (54) in a context in which the speaker knows Jo's height, we can slightly modify the lexical entry for *likely*, as in (60).

(60)
$$\overset{\textit{likely}}{\boxed{\begin{array}{l}\mathsf{do}\ \mathsf{background}_{\textit{likely}}\\ \quad cg \longleftarrow \mathtt{view}_{\mathrm{CG}}\\ \quad\boxed{(\lambda\phi.(\lambda d.(\lambda i'.Pr\left(\begin{array}{c}i'' \sim cg\\ \boxed{\phi(i_{w_{i''},c_{i'}})}\end{array}\right) \geq d)^{\eta})^{\eta})^{\eta}}\end{array}}} \vdash ap_{mod.}\backslash d/s$$

That is, *likely* passes the context part of the index it is evaluated at to its prejacent; meanwhile, it calculates the probability that its prejacent is true in the common ground by effectively marginalizing out the context parts of its indices, using only the world parts. Thus if there is no uncertainty about Jo's height in the context of an utterance of (54), the set of probabilities it characterizes at a given index will either be $\{1\}$ or $\{0\}$, depending on whether Jo's height is above or below that index's height threshold, respectively. Admittedly, this fact on its own does not explain the infelicity of (54) in such a context, but it at least appears to make way for the possibility of a pragmatic explanation.

Taking into account the updated lexical entry for *likely*, we can see in (61) the discourse that results from asking (55b).

(61)
```
do background_{how likely}
    $cg \leftarrow \mathtt{view}_{\mathrm{CG}}$
    background_{is it that}
    background_{Bo considers Jo tall}
    $qs \leftarrow \mathtt{view}_{\mathrm{QUD}}$
    $\mathtt{set}_{\mathrm{QUD}}(\langle \lambda d, \_.Pr\left(\begin{array}{l} i \sim cg \\ \boxed{\mathtt{height}(w_i)(\mathtt{j}) \geq \mathtt{d}_{tall}(\mathtt{c}(w_i)(\mathtt{b}))} \end{array}\right) \geq d, qs\rangle)$
```

Thus we can see that there are, in principle, two potential sources of occasional uncertainty about responses to the question asked in (61). First, there is uncertainty generated by world knowledge about Jo's height; the probability which *likely* computes across the indices of the common ground may, in fact, depend on this uncertainty. Crucially, second, there is also uncertainty generated by world knowledge about the height threshold which Bo considers to count as tall. Thus even if Jo's height is known in advance—and thus no uncertainty about it arises—the probability computed in (61) may nevertheless take on any value in the interval $[0, 1]$ in a way that depends on Bo.

5 Outlook

Here we take a step back to touch on some general points about the framework outlined, particularly regarding the way it respects the constraint of formal evaluability mentioned in Section 1. We especially expand on the claim that analyses in PDS may be compared with one another in terms of formally collected inference datasets. To demonstrate how this can be accomplished, we work with the simple discourse in (62), which features the *how likely* question of (54) from Section 4.3.

(62) a. Jo is an athlete.
b. How likely is it that Jo is tall?

Importantly, this discourse might naturally exist as a trial in an inference experiment employing (62b) as its question prompt. Because it is a degree question, a suitable response instrument might be a slider scale with labeled endpoints (in contrast to the question prompts invoked in Section 3.2.7). Moreover, appropriate labels might be, e.g., *not at all likely* on one end of the scale and *extremely likely* on the other end.

Let us assume that the trial represented in (62) gives rise to the discourse in (63).

```
(63)  $\mathsf{do}\ \mathsf{background}_{\textit{Jo is pred.-indef. athlete}}$
        $drs \leftarrow \mathsf{view}_{\mathsf{DRs}}$
        $cg \leftarrow \mathsf{view}_{\mathsf{CG}}$
        $\mathsf{set}_{\mathsf{CG}}\begin{pmatrix} i \sim cg \\ \mathsf{observe}(\mathsf{athlete}(w_i)(\mathrm{j})) \\ \boxed{i} \end{pmatrix}$
        $\mathsf{set}_{\mathsf{DRs}}(\lambda i, g.\{\langle \diamond, \mathrm{j}{::}g' \rangle \mid \langle \diamond, g' \rangle \in drs(i)(g) \land \mathsf{athlete}(w_i)(\mathrm{j})\})$
        $\mathsf{background}_{\textit{how likely}}$
        $cg' \leftarrow \mathsf{view}_{\mathsf{CG}}$
        $\mathsf{background}_{\textit{is it that Jo is tall}}$
        $qs \leftarrow \mathsf{view}_{\mathsf{QUD}}$
        $\mathsf{set}_{\mathsf{QUD}}(\langle \lambda d.Pr\begin{pmatrix} i' \sim cg' \\ \boxed{\mathsf{height}(w_{i'})(\mathrm{j}) \geq \mathrm{d}_{tall}(c_{i'})(\mathrm{j})} \end{pmatrix} \geq d), qs \rangle)$
```

If we assume—as is natural—that the program $\mathsf{background}_{\textit{how likely}}$ does not further modify the common ground after the preceding assertion, then (63) can be reduced somewhat by getting rid of the redundant view statement.

```
(64)  $\mathsf{do}\ \mathsf{background}_{\textit{Jo is pred.-indef. athlete}}$
        $drs \leftarrow \mathsf{view}_{\mathsf{DRs}}$
        $cg \leftarrow \mathsf{view}_{\mathsf{CG}}$
        $\mathsf{set}_{\mathsf{CG}}\begin{pmatrix} i \sim cg \\ \mathsf{observe}(\mathsf{athlete}(w_i)(\mathrm{j})) \\ \boxed{i} \end{pmatrix}$
        $\mathsf{set}_{\mathsf{DRs}}(\lambda i, g.\{\langle \diamond, \mathrm{j}{::}g' \rangle \mid \langle \diamond, g' \rangle \in drs(i)(g) \land \mathsf{athlete}(w_i)(\mathrm{j})\})$
        $\mathsf{background}_{\textit{how likely is it that Jo is tall}}$
        $qs \leftarrow \mathsf{view}_{\mathsf{QUD}}$
        $\mathsf{set}_{\mathsf{QUD}}(\langle \lambda d.Pr\begin{pmatrix} i \sim cg \\ \mathsf{observe}(\mathsf{athlete}(w_i)(\mathrm{j})) \\ \boxed{\mathsf{height}(w_i)(\mathrm{j}) \geq \mathrm{d}_{tall}(c_i)(\mathrm{j})} \end{pmatrix} \geq d), qs \rangle)$
```

Crucially, the information that Jo is an athlete is now transparently represented inside the scope of the probability operator which computes the degree interrogated by the pushed QUD.

Now let us take for granted the existence of a likelihood function $f_\Phi : r \rightarrow \mathsf{P}r$, whose range is a set of probability distributions over the closed unit interval $[0, 1]$ (i.e., the values of the slider scale). Then given some starting distribution bg over discourse states, we may invoke $\mathsf{respond}^{f_\phi}$, following Definition 3.30 of Section 3.2.7, applying it to (64) to obtain (65).

```
(65)  $s \sim bg$
      $\langle \diamond, s' \rangle \sim \mathsf{background}_{\textit{Jo is pred.-indef. athlete how likely is it that Jo is tall}}(s)$
      $i \sim \mathrm{CG}(s')$
      $\mathsf{observe}(\mathsf{athlete}(w_i)(\mathrm{j}))$
      $f(Pr\begin{pmatrix} i \sim \mathrm{CG}(s) \\ \mathsf{observe}(\mathsf{athlete}(w_i)(\mathrm{j})) \\ \boxed{\mathsf{height}(w_i)(\mathrm{j}) \geq \mathrm{d}_{tall}(c_i)(\mathrm{j})} \end{pmatrix}, \Phi)$
```

Let us dub (65) a *response model* for the experimental trial represented by (62).[19] When we model an inference dataset, however, we ultimately care about the parameters of a *statistical model* of the data and, in particular, how a distribution over those parameters relates to that data. In the case of the single trial represented in (65) as a discourse, a single datapoint will do: some or other value—call it 'y'—in the interval $[0, 1]$.

To take (65) onto a statistical model of y, we require one more piece of technology: density functions.[20]

Definition 5.1 (Density function)**.**

$$D_{(\cdot)} : \mathsf{P}\alpha \to \alpha \to r$$

Given a probabilistic program m that returns values of type α, $D_m : \alpha \to r$ is the function mapping any such value onto the density it is assigned by the probability distribution that m represents.

Now relating the response model in (65) to a statistical model of y involves a couple of minor syntactic changes. First, let us "absorb" the `background` program, along with the observation made about the common ground, into the starting distribution bg. Second, let us recover these aspects of (65) by *learning* what discourse effects they, in fact, have from the data y, using D. Finally, let us assume that both metalinguistic parameters *and* the nuisance parameters of the likelihood are encoded in bg, and that we simply return them. Making these changes yields (66).

(66)
$$\begin{array}{l} \langle s, \Phi\rangle \sim bg \\ inf \sim \boxed{\begin{array}{l} i \sim \mathsf{CG}(s) \\ \mathtt{observe}(\mathsf{athlete}(w_i)(\mathsf{j})) \\ \boxed{\mathsf{height}(w_i)(\mathsf{j}) \geq \mathsf{d}_{tall}(c_i)(\mathsf{j})} \end{array}} \\ \mathtt{factor}(D_{f(Pr(inf),\Phi)}(y)) \\ \boxed{\langle s, \Phi\rangle} \end{array}$$

Thus we now start by sampling parameters from the distribution bg; then we calculate the probability of a particular inference that depends on these parameters; and finally, we factor the weight associated with the parameters by the density that our likelihood—given the probability—assigns to y, before returning the parameters we started with.

Let us now say a little bit more about the background program bg itself. In the present context, bg acts as the prior distribution over the model's parameters. We might assume, for example, that the starting state parameterizes a Bernoulli distribution which decides whether or not Jo is an athlete, as well as two normal distributions determining Jo's height and the height threshold associated with the adjective *tall.* A normal distribution takes two parameters—a mean μ and a standard deviation $\sigma > 0$—and gives back a program of type $\mathsf{P}r$.

Definition 5.2 (Normal distribution)**.**

$$\mathcal{N} \quad : \quad r \times r \to \mathsf{P}r$$

[19] Our notion of a response model is analogous to that of a posterior predictive distribution employed in Bayesian statistics, insofar as such a model can be viewed as having been obtained from observed data.

[20] See Grove and Bernardy 2023 for a couple of approaches to implementing such functions in a continuation-based setting.

Furthermore, the Bernoulli distribution might, in turn, be parameterized by a Beta distribution, which given two parameters ($\alpha, \beta > 0$), computes values in the open interval $(0, 1)$.

Definition 5.3 (Beta distribution)**.**

$$\texttt{Beta} \ : \ r \times r \longrightarrow Pr$$

Meanwhile, the means of the normal distributions might themselves have normal priors, while the standard deviations might have as priors exponential distributions, which given a parameter $\lambda > 0$, compute non-negative values (with more mass assigned to values nearer to 0).

Definition 5.4 (Exponential distribution)**.**

$$\texttt{Exponential} \ : \ r \longrightarrow Pr$$

Putting all of this together yields the more spelled-out version of (66) in (67). Here, we continue to remain agnostic about the prior distribution over the nuisance parameters Φ. Note, also, the idiosyncratic use of notation for tuples in this example: we write, e.g., '$\text{CG} = m$' to indicate that m is the projection of the discourse state tuple corresonding to the common ground.[21]

$$
(67)\quad \begin{array}{l}
bg \sim \boxed{\begin{array}{l}
p_{\text{athlete}} \sim \texttt{Beta}(2,2)\\
\mu_{\text{height}} \sim \mathcal{N}(0,1)\\
\mu_{\mathrm{d}_{tall}} \sim \mathcal{N}(0,1)\\
\sigma_{\text{height}} \sim \texttt{Exponential}(1)\\
\sigma_{\mathrm{d}_{tall}} \sim \texttt{Exponential}(1)\\
\Phi \sim \texttt{NuisancePrior}\\
\boxed{\langle \text{CG} = \left(\begin{array}{l} \tau \sim \texttt{Bernoulli}(p_{\text{athlete}})\\ h \sim \mathcal{N}(\mu_{\text{height}}, \sigma_{\text{height}})\\ d \sim \mathcal{N}(\mu_{\mathrm{d}_{tall}}, \sigma_{\mathrm{d}_{tall}})\\ \boxed{\langle \text{athlete} = \lambda x.\tau, \text{height} = \lambda x.h, \mathrm{d}_{tall} = \lambda x.d\rangle}\end{array}\right), \Phi\rangle}
\end{array}}\\
\langle s, \Phi\rangle \sim bg\\
inf \sim \boxed{\begin{array}{l}
i \sim \text{CG}(s)\\
\texttt{observe}(\text{athlete}(w_i)(\text{j}))\\
\boxed{\text{height}(w_i)(\text{j}) \geq \mathrm{d}_{tall}(c_i)(\text{j})}
\end{array}}\\
\texttt{factor}(D_{f(Pr(inf),\Phi)}(y))\\
\boxed{\langle s, \Phi\rangle}
\end{array}
$$

At the end of the day, we have a probability distribution over metalinguistic parameters which has been fit to y. This is our statistical model.

To round out the discussion, let's make this view somewhat more abstract. Say (68) is the trial in which experimental item j (where $1 \leq j \leq n_{\text{item}}$) is presented to experimental participant p (where $1 \leq p \leq n_{\text{participant}}$).

[21]Our use of a $\texttt{Beta}(2, 2)$ prior for the Bernoulli parameter means that somewhat more weight is given to probabilities nearer to 0.5 and somewhat less weight is given to probabilities nearer to 0 and 1.

(68) a. $S_{j,p}$

b. $Q_{j,p}$?

Moreover, assume the dataset $\boldsymbol{y}_{j,p}$ gives datapoints for each such trial. The model for (68) is then defined in (69), where $\texttt{S}_{j,p}$ and $\texttt{Q}_{j,p}$ provide the probabilistic dynamic semantic values of (68a) and (68b), respectively.

(69)
$$\begin{array}{l}
\langle ..., \sigma_{j,p}, \Phi_{j,p}, ... \rangle \sim bg \\
\vdots \\
\langle \diamond, \sigma'_{j,p} \rangle \sim \texttt{assert}(\texttt{S}_{j,p})(\sigma_{j,p}) \\
\langle \diamond, \sigma''_{j,p} \rangle \sim \texttt{ask}(\texttt{Q}_{j,p})(\sigma'_{j,p}) \\
i_{j,p} \sim \mathsf{CG}(\sigma''_{j,p}) \\
\texttt{factor}(D_{f(\mathsf{max}(\lambda x.\mathsf{QUD}(\sigma''_{j,p})_0(x)(i_{j,p})),\Phi_{j,p})}(\boldsymbol{y}_{j,p})) \\
\vdots \\
\boxed{\langle ..., \sigma_{j,p}, \Phi_{j,p}, ... \rangle}
\end{array}$$

While (69) presents the most general form of such a model, one can make various simplifying assumptions. For example, the initial discourse states $\sigma_{j,p}$ may not be distributed independently across participants—or even items, for that matter—but may rather share certain structure. Indeed, any such simplifications can be viewed as being determined by the prior bg. Importantly, this presentation can be adapted to a Bayesian setting, in which bg may have somewhat rich structure, and it is the full posterior distribution defined in (69) which is assigned significance; alternatively, one may potentially take a more frequentist approach by making bg relatively "flat" and estimating the mode of (69). There may be a certain philosophical benefit, generally speaking, of going with the Bayesian approach, however. One can view the distribution encoded in (69)—or perhaps one of the marginal distributions gotten by fixing j and p—as representing a discourse participant's cognitive state. Specifically, this distribution represents their knowledge about the events of the discourse—which assertions were made and which questions were asked—but also, their rich lexical knowledge about the meanings of the expressions used.

Indeed, one may adopt different starting assumptions about the metalinguistic parameters involved in an analysis of some data—in particular, how they encode formal aspects of lexical meaning. The upshot is that such differences result in different statistical models, which may be compared using standard comparison metrics. It is in this way that PDS provides a means of both developing and comparing semantic theories.

6 Conclusion

We have introduced Probabilistic Dynamic Semantics (PDS) as a framework that merges the strengths of Dynamic Montague Semantics (DMS) with probabilistic reasoning. PDS maintains the compositional and modular characteristics that make modern DMS powerful, while introducing probabilistic elements that allow for more nuanced predictions about semantic inference. By retaining the core structure of traditional dynamic semantic representations and augmenting them with representations of probabilistic programs, PDS offers a dual-level analysis: one level which preserves the interpretative richness of DMS and another enabling precise, quantitative assessment in terms of experimental data. This approach not only allows for the seamless incorporation of probabilistic reasoning into existing semantic theories, but also facilitates the explicit modeling of how linguistic expressions are interpreted in context, considering both the range of possible meanings and the likelihood of those meanings being inferred by language comprehenders.

Our exposition has been primarily formal, laying the groundwork for future empirical investigations. The ability to develop, test, and refine semantic analyses using experimentally derived datasets is integral to the advancement of semantic theory—we hope that PDS can support this goal.

References

Asher, Nicholas. 1993. *Reference to Abstract Objects in Discourse.* Ed. by Gennaro Chierchia, Pauline Jacobson, and Francis J. Pelletier. Vol. 50. Studies in Linguistics and Philosophy. Dordrecht: Springer Netherlands. DOI: 10.1007/978-94-011-1715-9.

Atkey, Robert. 2009. Parameterised notions of computation. *Journal of Functional Programming* 19.3-4, pp. 335–376. DOI: 10.1017/S095679680900728X.

Barker, Chris. 2002. The Dynamics of Vagueness. *Linguistics and Philosophy* 25.1, pp. 1–36. DOI: 10.1023/A:1014346114955.

Barker, Chris and Chung-chieh Shan. 2014. *Continuations and natural language.* Vol. 53. Oxford studies in theoretical linguistics.

Beaver, David I. 1999. Presupposition Accomodation: A Plea for Common Sense. In *Logic, language, and computation.* Ed. by Lawrence S. Moss, Jonathan Ginzburg, and Rijke de Maarten. Vol. 2. Stanford: CSLI Publications, pp. 21–44.

Beaver, David I. 2001. *Presupposition and Assertion in Dynamic Semantics.* Studies in Logic, Language and Information. Stanford: CSLI Publications.

Benton, P. N., G. M. Bierman, and V. C. V. De Paiva. 1998. Computational types from a logical perspective. *Journal of Functional Programming* 8.2, pp. 177–193. DOI: 10.1017/S0956796898002998.

Bergen, Leon, Roger Levy, and Noah Goodman. 2016. Pragmatic reasoning through semantic inference. *Semantics and Pragmatics* 9, ACCESS–ACCESS. DOI: 10.3765/sp.9.20.

Bernardy, Jean-Philippe, Rasmus Blanck, Stergios Chatzikyriakidis, Shalom Lappin, et al. 2019. Predicates as Boxes in Bayesian Semantics for Natural Language. *Proceedings of the 22nd Nordic Conference on Computational Linguistics.* Turku, Finland: Linköping University Electronic Press, pp. 333–337.

Bernardy, Jean-Philippe, Rasmus Blanck, Stergios Chatzikyriakidis, and Aleksandre Maskharashvili. 2022. Bayesian Natural Language Semantics and Pragmatics. In *Probabilistic Approaches to Linguistic Theory.* Ed. by Jean-Philippe Bernardy et al. CSLI Publications.

Bumford, Dylan and Simon Charlow. 2022. *Dynamic semantics with static types.*

Charlow, Simon. 2014. On the semantics of exceptional scope. PhD Thesis. New York: New York University.

Charlow, Simon. 2019. Where is the destructive update problem? *Semantics and Pragmatics* 12, 10:1–24. doi: 10.3765/sp.12.10.

Charlow, Simon. 2020a. Static and dynamic exceptional scope.

Charlow, Simon. 2020b. The scope of alternatives: indefiniteness and islands. *Linguistics and Philosophy* 43.4, pp. 427–472. doi: 10.1007/s10988-019-09278-3.

Coppock, Elizabeth and David Beaver. 2015. Definiteness and Determinacy. *Linguistics and Philosophy* 38.5, pp. 377–435.

De Groote, Philippe. 2006. Towards a Montagovian Account of Dynamics. *Semantics and Linguistic Theory* 16.0, pp. 1–16. doi: 10.3765/salt.v16i0.2952.

Dekker, Paul. 1993. Transsentential Meditations; Ups and downs in dynamic semantics. doctoral. University of Amsterdam.

Dekker, Paul. 1994. Predicate Logic with Anaphora. *Proceedings of the 4th Conference on Semantics and Linguistic Theory*. Ed. by Mandy Harvey and Lynn Santelmann. Ithaca, NY: Cornell, pp. 79–95.

Elliott, Patrick D. 2022. A flexible scope theory of intensionality. *Linguistics and Philosophy*. doi: 10.1007/s10988-022-09367-w.

Erk, Katrin. 2022. The Probabilistic Turn in Semantics and Pragmatics. *Annual Review of Linguistics* 8.Volume 8, 2022, pp. 101–121. doi: 10.1146/annurev-linguistics-031120-015515.

Erk, Katrin and Aurélie Herbelot. 2023. How to Marry a Star: Probabilistic Constraints for Meaning in Context. *Journal of Semantics* 40.4, pp. 549–583. doi: 10.1093/jos/ffad016.

Farkas, Donka F. and Kim B. Bruce. 2010. On Reacting to Assertions and Polar Questions. *Journal of Semantics* 27.1, pp. 81–118. doi: 10.1093/jos/ffp010.

Frank, Michael C. and Noah D. Goodman. 2012. Predicting Pragmatic Reasoning in Language Games. *Science* 336.6084, pp. 998–998. doi: 10.1126/science.1218633.

Ginzburg, Jonathan. 1996. Dynamics and the semantics of dialogue. In *Logic, Language, and Computation*. Ed. by Jerry Seligman and Dag Westerståhl. Vol. 1. Stanford: CSLI Publications, pp. 221–237.

Giorgolo, Gianluca and Ash Asudeh. 2014. One Semiring to Rule Them All. *CogSci 2014 Proceedings*.

Goodman, Noah D. and Daniel Lassiter. 2015. Probabilistic Semantics and Pragmatics Uncertainty in Language and Thought. In *The Handbook of Contemporary Semantic Theory*. Ed. by Shalom Lappin and Chris Fox. John Wiley & Sons, Ltd, pp. 655–686. doi: 10.1002/9781118882139.ch21.

Groenendijk, Jeroen and Martin Stokhof. 1984. Studies on the semantics of questions and the pragmatics of answers. PhD thesis. Amsterdam: University of Amsterdam.

Groenendijk, Jeroen and Martin Stokhof. 1990. Dynamic Montague Grammar. In *Proceedings of the Second Symposium on Logic and Language*. Ed. by László Kálmán and László Pólos. Budapest: Eötvös Loránd Press, pp. 3–48.

Groenendijk, Jeroen A. and Martin B. J. Stokhof. 1991. Dynamic Predicate Logic. *Linguistics and Philosophy* 14.1, pp. 39–100.

Grove, Julian. 2019. Scope-taking and presupposition satisfaction. PhD Thesis. Chicago: University of Chicago.

Grove, Julian. 2022. Presupposition projection as a scope phenomenon. *Semantics and Pragmatics* 15, 15:1–39. DOI: 10.3765/sp.15.15.

Grove, Julian and Jean-Philippe Bernardy. 2023. Probabilistic Compositional Semantics, Purely. *New Frontiers in Artificial Intelligence*. Ed. by Katsutoshi Yada et al. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, pp. 242–256. DOI: 10.1007/978-3-031-36190-6_17.

Grove, Julian and Aaron Steven White. 2024. *Factivity, presupposition projection, and the role of discrete knowlege in gradient inference judgments*.

Hamblin, C. L. 1973. Questions in Montague English. *Foundations of Language* 10.1, pp. 41–53.

Hausser, Roland and Dietmar Zaefferer. 1978. Questions and Answers in a Context-Dependent Montague Grammar. In *Formal Semantics and Pragmatics for Natural Languages*. Ed. by F. Guenthner and S. J. Schmidt. Dordrecht: Springer Netherlands, pp. 339–358. DOI: 10.1007/978-94-009-9775-2_12.

Hausser, Roland R. 1983. The Syntax and Semantics of English Mood. In *Questions and Answers*. Ed. by Ferenc Kiefer. Dordrecht: Springer Netherlands, pp. 97–158. DOI: 10.1007/978-94-009-7016-8_6.

Heim, Irene. 1982. The Semantics of Definite and Indefinite Noun Phrases. PhD Thesis. Amherst: University of Massachussetts.

Heim, Irene. 1983. File Change Semantics and the Familiarity Theory of Definiteness. In *Meaning, Use, and Interpretation of Language*. Ed. by Rainer Bäuerle, Christoph Schwarze, and Arnim von Stechow. De Gruyter, pp. 164–189. DOI: 10.1515/9783110852820.164.

Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Malden, MA: Blackwell.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*. Ed. by Arnold M. Zwicky, David R. Dowty, and Lauri Karttunen. Studies in Natural Language Processing. Cambridge: Cambridge University Press, pp. 206–250. DOI: 10.1017/CBO9780511597855.007.

Kamp, Hans. 1981. A Theory of Truth and Semantic Representation. In *Formal Methods in the Study of Language*. Ed. by Jeroen A. Groenendijk, Theo M. V. Janssen, and Martin B. J. Stokhof. Mathematical Centre Tracts 135. Amsterdam: Mathematisch Centrum, pp. 277–322.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Vol. 42. Studies in Linguistics and Philosophy. Dordrecht: Springer.

Karttunen, Lauri. 1976. Discourse Referents. In *Notes from the Linguistic Underground*. Brill, pp. 363–385. DOI: 10.1163/9789004368859_021.

Karttunen, Lauri. 1977. Syntax and Semantics of Questions. *Linguistics and Philosophy* 1.1, pp. 3–44.

Kennedy, Christopher. 2007. Vagueness and grammar: the semantics of relative and absolute gradable adjectives. *Linguistics and Philosophy* 30.1, pp. 1–45. DOI: 10.1007/s10988-006-9008-0.

Kennedy, Christopher and Louise McNally. 2005. Scale Structure, Degree Modification, and the Semantics of Gradable Predicates. *Language* 81.2, pp. 345–381. DOI: 10.1353/lan.2005.0071.

Kobele, Gregory. 2006. Generating Copies: An investigation into structural identity in language and grammar. PhD thesis. Los Angeles: UCLA.

Lassiter, Daniel. 2011. Vagueness as Probabilistic Linguistic Knowledge. *Vagueness in Communication*. Ed. by Rick Nouwen et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 127–150. DOI: 10.1007/978-3-642-18446-8_8.

Lassiter, Daniel and Noah D. Goodman. 2017. Adjectival vagueness in a Bayesian model of interpretation. *Synthese* 194.10, pp. 3801–3836. DOI: 10.1007/s11229-015-0786-1.

Liang, Shen, Paul Hudak, and Mark Jones. 1995. Monad transformers and modular interpreters. *POPL '95 Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. New York, pp. 333–343.

Muskens, Reinhard. 1996. Combining Montague semantics and discourse representation. *Linguistics and Philosophy* 19.2, pp. 143–186. DOI: 10.1007/BF00635836.

Phillips, Colin et al. 2021. Theories All the Way Down: Remarks on "Theoretical" and "Experimental" Linguistics. In *The Cambridge Handbook of Experimental Syntax*. Ed. by Grant Goodall. Cambridge Handbooks in Language and Linguistics. Cambridge: Cambridge University Press, pp. 587–616. DOI: 10.1017/9781108569620.023.

Roberts, Craige. 2012. Information Structure: Towards an integrated formal theory of pragmatics. *Semantics and Pragmatics* 5, 6:1–69. DOI: 10.3765/sp.5.6.

Shan, Chung-chieh. 2002. Monads for natural language semantics. *arXiv:cs/0205026*.

Shan, Chung-chieh. 2004. Binding Alongside Hamblin Alternatives Calls for Variable-free Semantics. *Semantics and Linguistic Theory* 14.0, pp. 289–304. DOI: 10.3765/salt.v14i0.2901.

Simons, Mandy. 2007. Observations on embedding verbs, evidentiality, and presupposition. *Lingua* 117.6, pp. 1034–1056. DOI: 10.1016/j.lingua.2006.05.006.

Simons, Mandy, David Beaver, et al. 2017. The Best Question: Explaining the Projection Behavior of Factives. *Discourse Processes* 54.3, pp. 187–206.

Simons, Mandy, Judith Tonhauser, et al. 2010. What projects and why. *Semantics and Linguistic Theory*. Ed. by Nan Li and David Lutz. Vol. 20. University of British Columbia and Simon Fraser University: Linguistic Society of America, pp. 309–327. DOI: 10.3765/salt.v20i0.2584.

Stalnaker, Robert. 1978. Assertion. In *Pragmatics*. Ed. by Peter Cole. Vol. 9. New York: Academic Press, pp. 315–332.

Steedman, Mark. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monographs. Cambridge: MIT Press.

Steedman, Mark. 2000. *The Syntactic Process*. Cambridge: MIT Press.

Tonhauser, Judith, David I. Beaver, and Judith Degen. 2018. How Projective is Projective Content? Gradience in Projectivity and At-issueness. *Journal of Semantics* 35.3, pp. 495–542. DOI: 10.1093/jos/ffy007.

Unger, Christina. 2012. Dynamic Semantics as Monadic Computation. *New Frontiers in Artificial Intelligence*. Ed. by Manabu Okumura, Daisuke Bekki, and Ken Satoh. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 68–81. DOI: 10.1007/978-3-642-32090-3_7.

Van Benthem, Johan, Jelle Gerbrandy, and Barteld Kooi. 2009. Dynamic Update with Probabilities. *Studia Logica* 93.1, pp. 67–96. DOI: 10.1007/s11225-009-9209-y.

Vijay-Shanker, K. and D. J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical systems theory* 27.6, pp. 511–546. DOI: 10.1007/BF01191624.

Wadler, Philip. 1992. Comprehending Monads. In *Mathematical Structures in Computer Science*. Vol. 2. Cambridge University Press, pp. 461–493.

Xiang, Yimei. 2021. A hybrid categorial approach to question composition. *Linguistics and Philosophy* 44.3, pp. 587–647. DOI: 10.1007/s10988-020-09294-8.

Zeevat, Henk. 2013. Implicit Probabilities in Update Semantics. In *The dynamic, inquisitive, and visionary life of ϕ, $?\phi$, and $\diamond\phi$: A festschrift for Jeroen Groenendijk, Martin Stokhof, and Frank Veltman*. Amsterdam: ILLC, pp. 319–324.