

Linguistics 341

Spring 2025

PROBLEM SET 4

Due: Wednesday, April 6, 6:00 PM

Aidan Mokalla

Compositional interpretation of definite DPs

- [i] (a) $\llbracket \text{the} \rrbracket^g = \lambda P[x : \forall y : P(y) \rightarrow (x = y)] \in D_{\langle \langle e, t \rangle, e \rangle}$
 (b) $\llbracket \text{black} \rrbracket^g = \lambda x[\text{black}'(x)] \in D_{\langle e, t \rangle}$
 (c) $\llbracket \text{kitten} \rrbracket^g = \lambda x[\text{kitten}'(x)] \in D_{\langle e, t \rangle}$
 (d) $\llbracket \text{Alex} \rrbracket^g = \text{Alex}' \in D_e$
 (e) $\llbracket \text{adopted} \rrbracket^g = \lambda y. \lambda x[\text{adopt}'(x, y)] \in D_{\langle e, \langle e, t \rangle \rangle}$
 (f) $\llbracket t_a \rrbracket^g = t_a' \in D_e$

[ii] (a)
$$\left[\begin{array}{c} \text{VP} \\ \swarrow \quad \searrow \\ \text{V} \quad \text{DP}_4 \\ | \quad | \\ \text{adopted} \quad t_a \end{array} \right]^g \stackrel{\text{FA}}{=} \lambda y. \lambda x[\text{adopt}'(x, y)](t_a')$$

$$\Rightarrow \lambda x[\text{adopt}'(x, t_a')] \in D_{\langle e, t \rangle}$$

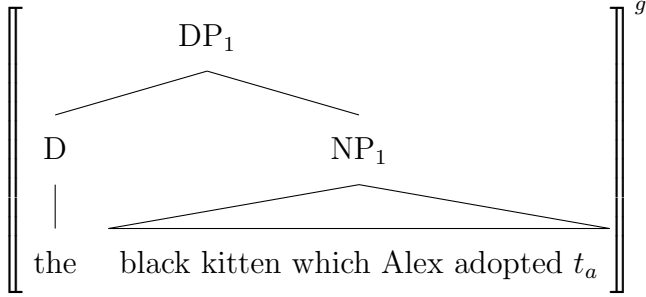
$$\begin{array}{c}
\text{(b)} \quad \left[\left[\begin{array}{c} \text{S} \\ \swarrow \quad \searrow \\ \text{DP}_3 \quad \text{VP} \\ | \quad \swarrow \quad \searrow \\ \text{Alex} \quad \text{adopted } t_a \end{array} \right] \right]^g \stackrel{\text{FA}}{=} \lambda x [\text{adopt}'(x, t_a')](\text{Alex}') \\
\Rightarrow \text{adopt}'(\text{Alex}', t_a') \in D_t
\end{array}$$

$$\begin{array}{c}
\text{(c)} \quad \left[\left[\begin{array}{c} \text{CP} \\ \swarrow \quad \searrow \\ \text{DP}_2 \quad \text{C}' \\ | \quad \swarrow \quad \searrow \\ \text{which}_a \quad \text{C} \quad \text{S} \\ \swarrow \quad \searrow \\ \text{Alex adopted } t_a \end{array} \right] \right]^g \stackrel{\text{PA}}{=} \lambda z \left[\left[\begin{array}{c} \text{C}' \\ \swarrow \quad \searrow \\ \text{C} \quad \text{S} \\ \swarrow \quad \searrow \\ \text{Alex adopted } t_a \end{array} \right] \right]^{g^{z/a}} \\
\stackrel{\text{C vacuity}}{=} \lambda z \left[\left[\begin{array}{c} \text{S} \\ \swarrow \quad \searrow \\ \text{Alex adopted } t_a \end{array} \right] \right]^{g^{z/a}} \stackrel{[\text{ii}](\text{b})}{=} \lambda z [\text{adopt}'(\text{Alex}', z)] \in D_{\langle e, t \rangle}
\end{array}$$

$$\begin{array}{c}
\text{(d)} \quad \left[\left[\begin{array}{c} \text{NP}_2 \\ \swarrow \quad \searrow \\ \text{NP}_3 \quad \text{CP} \\ | \quad \swarrow \quad \searrow \\ \text{kitten} \quad \text{which Alex adopted } t_a \end{array} \right] \right]^g \stackrel{\text{PM}}{=} \lambda z [\llbracket \text{kitten} \rrbracket(z) \wedge \llbracket \text{which Alex adopted } t_a \rrbracket(z)] \\
\stackrel{[\text{i}](\text{c}), [\text{ii}](\text{c})}{=} \lambda z [\text{kitten}'(z) \wedge \text{adopt}'(\text{Alex}', z)] \in D_{\langle e, t \rangle}
\end{array}$$

$$\text{(e)} \quad \left[\left[\begin{array}{c} \text{NP}_1 \\ \swarrow \quad \searrow \\ \text{AP} \quad \text{NP}_2 \\ | \quad \swarrow \quad \searrow \\ \text{black} \quad \text{kitten which Alex adopted } t_a \end{array} \right] \right]^g$$

$$\begin{aligned}
& \stackrel{\text{PM}}{=} \lambda z [\llbracket \text{black} \rrbracket(z) \wedge \llbracket \text{kitten which Alex adopted } t_a \rrbracket] \\
& \stackrel{[\text{i}](\text{b}), [\text{ii}](\text{d})}{=} \lambda z [\text{black}'(z) \wedge \text{kitten}'(z) \wedge \text{adopt}'(\text{Alex}', z)] \in D_{\langle e, t \rangle} \\
\end{aligned}$$

(f) 

$$\begin{aligned}
& \stackrel{\text{FA}}{=} \lambda P[x : \forall y : P(y) \rightarrow (x = y)](\lambda z [\text{black}'(z) \wedge \text{kitten}'(z) \wedge \text{adopt}'(\text{Alex}', z)]) \\
& \Rightarrow x : \forall y : \lambda z [\text{black}'(z) \wedge \text{kitten}'(z) \wedge \text{adopt}'(\text{Alex}', z)](y) \rightarrow (x = y) \\
& \Rightarrow x : \forall y : (\text{black}'(y) \wedge \text{kitten}'(y) \wedge \text{adopt}'(\text{Alex}', y)) \rightarrow (x = y) \quad \square
\end{aligned}$$

Modeling sentences with quantifiers using set theory

- (a) $\llbracket \text{six NP VP} \rrbracket^c = 1$ iff $|\text{NP} \cap \text{VP}| \geq 6$
- (b) $\llbracket \text{both (of the) NP VP} \rrbracket^c = 1$ iff $|\text{NP}| = 2 \wedge \text{NP} \subseteq \text{VP}$
- (c) The necessary and sufficient conditions for $\llbracket \text{many NP VP} \rrbracket^c = 1$ are seemingly difficult and complicated to express with the tools we've investigated thus far. To address this, I'm going to suggest a small expansion to our theory. Since this expansion allows us to interact with new concepts, I'll also introduce new notation to represent these concepts.

To start with, consider how my response above to question (b) needed the “ \wedge ” symbol, which hails from logic rather than from pure set theory. Before that, I needed the symbols “1,” “2”, and “ \geq ”, which hail from arithmetic instead of from pure set theory. I could have instead used “pure” set theory to write that

$$\begin{aligned}
& \exists f : \llbracket \text{both (of the) NP VP} \rrbracket^c \leftrightarrow \{\emptyset\} \\
& \Leftrightarrow \exists g : \text{NP} \leftrightarrow \{\emptyset, \{\emptyset\}\}, \exists h : \text{NP} \rightarrow \text{VP} : \forall x \in \text{NP} : h(x) = x.
\end{aligned}$$

However, it's not clear whether a strict adherence to set theoretic notation truly helps us capture our intuition about what's happening here. Is the best way to describe what “both” means really “there exists a bijective function f between the denotation of ‘both NP VP’ and the fundamental set of cardinality 1 iff there exists a bijective function g between NP and the fundamental set of cardinality 2 and there exists a function $h : \text{NP} \rightarrow \text{VP}$ such that for all $x \in \text{NP}$, $h(x) = x$ ”? I'd say no, and that we should probably add some more machinery to our model since it would allow us to more accurately and concisely describe what we're trying to describe. My stance is that adding “unnecessary” complexity to our model is more warranted in cases where the utility of the helpful parts of the new machinery (e.g. arithmetic numerals, inequalities) outweighs the additional baggage we must accept along with the helpful parts. Does our semantics really make use of an *entire* arithmetic, e.g. \div , π , etc., or only the parts that are useful for our purposes?

Set theory has brought us very far, and its simple constructions have proven useful. As our model of natural language semantics has grown more advanced this semester, we have already allowed Occam's razor to dull so that certain crevices may be filled. Lambda calculus, like many axiomatic systems, is based on set theory, and was our most recent natural extension of set theory to the semantics of natural languages, preceded by predicate logic (and arithmetic before that). An obvious candidate for the next natural extension to our set theoretic foundations is probability theory, which like its predecessors, has set theoretic foundations.

Currently, we classify our denotations into the following recursively defined types:

$$\begin{aligned} D_t &= \{0, 1\} \\ D_e &= \{e_1, e_2, e_3, \dots\} \\ D_{\langle e, t \rangle} &= D_e \times D_t \\ &\dots \end{aligned}$$

$$D_{\langle a,b \rangle} = D_a \times D_b$$

We understood the denotation of NPs and certain VPs as a function from elements of D_e to either 0 or 1, and called this relation *function application*:

$$\llbracket x_{\langle a,b \rangle} \rrbracket(y_a) \mapsto \begin{cases} z \in \{0, 1\} = D_t & \text{if } x \in D_{\langle a,t \rangle} \\ z \in D_{\langle e,t \rangle} & \text{if } x \in D_{\langle a,\langle e,t \rangle \rangle} \\ \dots & \dots \\ z \in D_b & \text{in general} \end{cases}$$

Now, I can explain precisely what the change I want to propose is. Currently, we understand the core of our semantic system to be D_e (the set of all entities) and D_t (the set containing 0 and 1). We should not be concerned that D_e is quite large, since it serves as one of the two fundamental structural elements of our theory. Conversely, it's surprising that the other necessary piece of structure in our theory (i.e. D_t) be so simple.

If we expand one of our recursive base cases slightly, and instead allow every value between 0 and 1 to compose D_t , then we can reinterpret our type system with minimal changes as

$$D_{\textcolor{red}{t}} = [0, 1]$$

$$D_e = \{e_1, e_2, e_3, \dots\}$$

$$D_{\langle e,t \rangle} = D_e \times D_{\textcolor{red}{t}}$$

$$\dots$$

$$D_{\langle a,b \rangle} = D_a \times D_b$$

Every predicate $P \in D_{\langle e,t \rangle}$ can now be understood as a function that maps all entities to a value *between* (or at) 0 and 1,

$$P_{\langle e,t \rangle} : D_e \rightarrow [\textcolor{red}{0}, \textcolor{red}{1}],$$

instead of its old definition of $P_{\langle e,t \rangle} : D_e \rightarrow \{0, 1\}$. This is not to say that the

truth values of all statements lay between 0 and 1. Many of them *are* exactly 0 or exactly 1. However, by allowing a limited number of statements to lie in this third area, we may reap a slew of theoretical desiderata. To start with, how might this change affect our construction of more complex types? Aside from the expansion of our base type from D_t to $D_{\textcolor{red}{t}}$, our type system needn't change further:

$$\llbracket x_{\langle a, b \rangle} \rrbracket(y_a) \mapsto \begin{cases} z \in [0, 1] = D_{\textcolor{red}{t}} & \text{if } x \in D_{\langle a, \textcolor{red}{t} \rangle} \\ z \in D_{\langle e, \textcolor{red}{t} \rangle} & \text{if } x \in D_{\langle a, \langle e, \textcolor{red}{t} \rangle \rangle} \\ \dots & \dots \\ z \in D_b & \text{in general} \end{cases}$$

Before fleshing out the rest of the details, let's consider explanations of how we arrive at the denotation of “the black cat” in both of these models. Starting with our current model, we might derive the following, with $c(D_e)$ representing all entities in the context c (i.e. the model $M = c(D_e) \in c$):

$$\begin{aligned} \llbracket \text{the black cat} \rrbracket^c &= \lambda X_{\langle e, t \rangle}. \lambda Y_{\langle e, t \rangle} [z : \forall w \in c(D_e) : (Y(w) \wedge X(w)) \leftrightarrow w = z] (\text{cat}')(\text{black}') \\ &\Rightarrow \lambda Y_{\langle e, t \rangle} [z : \forall w \in c(D_e) : (Y(w) \wedge \text{black}'(w)) \leftrightarrow w = z] (\text{cat}') \\ &\Rightarrow z : \forall w \in c(D_e) : (\text{cat}'(w) \wedge \text{black}'(w)) \leftrightarrow w = z \\ &\in c(D_e) \subseteq D_e \end{aligned}$$

Notice that with this interpretation, our theory must ask us to assume that each time the word “the” is used, the linguistic system is able to perform this final step above. That is, we must believe that given some arbitrary predicate P , we are able to efficiently determine the unique $z \in c(D_e)$ such that $\forall w \in c(D_e) : P(w) \leftrightarrow w = z$ in a consistent amount of time. The exact manner in which we determine this z is thus an important question—until now, we have assumed that when z is described to the semantic system as the unique satisfying argument of some predicate, regardless of how complex that predicate may be, the exact identity of z (i.e. $\llbracket z \rrbracket$) simply springs into in the mind.

One thing we know for certain about how we determine which entity z is, is

that every candidate in the context (or if we prefer, without loss of generality, every possible context) must be considered at some point. This explanation requires us to conclude that every possible candidate is checked because in order to verify that there is only one unique z in the maximal domain of discourse that satisfies the predicate P , we cannot leave any entity unturned, not matter how unlikely it is to satisfy P . Carrying this line of reasoning further, we have exactly two possibilities:

- (1) We cannot consider more than one candidate for z simultaneously.

In this case, we must consider every candidate from the set of all entities in the domain of discourse ($c(D_e)$) sequentially, although each individual evaluation may be quite fast. In what order, then, do we evaluate candidates? Do we choose randomly? If so, how do we remember which candidates we've evaluated and which we haven't? Or, do we evaluate according to some predefined ordering? If so, what is this ordering determined by?

- (2) We can consider more than one candidate for z simultaneously.

In this case, I would have similar questions. Why does this specific facet of the linguistic system allow for blatant parallelism, when most of our theory is built around binary sequential operations like merge instead of simultaneous repetition of the same operation with different arguments? It seems impossible that we're capable of considering an arbitrarily high number of candidates simultaneously, so there must exist some upper bound on how many candidates can be evaluated at once. When we need to evaluate more than this number of candidates, how do we decide which to evaluate first?

In either case, other questions remain as well. For example, why is it that even when $|c_1(D_e)|$ is much smaller than $|c_2(D_e)|$, it takes us about the same amount of time to understand the meaning of $\llbracket \text{the NP}_i \rrbracket^{c_1}$ and $\llbracket \text{the NP}_i \rrbracket^{c_2}$? Shouldn't we expect to be able to understand the meaning of $\llbracket \text{the NP}_i \rrbracket^{c_1}$ much more quickly than the meaning of $\llbracket \text{the NP}_i \rrbracket^{c_2}$, since the number of possible candidates z for satisfying $\text{NP}'_i(z) = 1$ is exponentially smaller, making it easier for us to check each of them? Perhaps, I hear you saying, "there actually

is a difference in how long it takes!” However, if $|c_2| - |c_1| = n$, and a listener takes t seconds to understand the meaning of $\llbracket \text{the NP}_i \rrbracket^{c_1}$, then we should expect that it take a listener $\mathcal{O}(2^n) \times t$ seconds to understand the meaning of $\llbracket \text{the NP}_i \rrbracket^{c_2}$. Does the prediction that it take exponentially longer to understand a statement in a linearly larger context, which is predicted by our current binary model of semantic truth, match your intuition?

Notice also how we can have a nested series of this type of operation, such as in the DP “the black cat that was sitting on the sidewalk next to the Jacksons’ house on the day before yesterday with the...” This means that we’d need to determine which entity uniquely satisfies all of these predicates at the same time, and would have to consider a hyper-exponential number of candidates. In doing so, you’d also be completing the simpler task of determining whether there even exists an entity that has all of these properties in the first place, i.e. whether there is a collection of non-contradictory binary properties that **satisfy** these predicates (as well as whether such an entity is in the domain of discourse). Simply showing that it’s possible for someone to determine whether such an entity even exists, without evaluating every single possible entity, would be equivalent to showing that the **SAT** problem is solvable in polynomial time. Putting aside an explanation of what exactly this means, which is not important, such a demonstration of how this is possible would directly and concretely entail that $P=NP$, that you would be the first person eligible for a \$1,000,000 cash prize, that it’s equally difficult for people to play Sudoku as it is for them to sort a list of numbers from lowest to highest, and that you would be remembered throughout history as proving a something that thousands have spent decades trying to disprove.

I argue that our current model of semantic evaluation would force us to assume that certain highly unlikely and discredited properties of our universe are true. A much more defensible approach is an explanation that first somehow weeds out candidates that are unlikely to satisfy the predicate before they are even considered in the first place. Such a heuristic would allow us to use our time more efficiently by first evaluating only those candidates we believe *a priori*

are actually likely to perhaps satisfy the predicate(s).

In our new model, we can simultaneously simplify the semantics of “the AP NP” and resolve the types of questions above, by instead defining the its denotation as the “maximally AP and maximally NP entity”:

$$\begin{aligned}
\llbracket \text{the black cat} \rrbracket^c &= \lambda X_{\langle e, t \rangle} . \lambda Y_{\langle e, t \rangle} \left[\arg \max_{z \in c(D_e)} Y(z) \times X(z) \right] (\text{cat}')(\text{black}') \\
&= \lambda Y_{\langle e, t \rangle} \left[\arg \max_{z \in c(D_e)} Y(z) \times \text{black}'(z) \right] (\text{cat}') \\
&= \arg \max_{z \in c(D_e)} \text{cat}'(z) \times \text{black}'(z) \\
&\in c(D_e) \subseteq D_e
\end{aligned}$$

We have made two changes: we were able to simplify the the uniqueness requirement to an “arg max,” and we changed the binary operation that defines predicate modification from \wedge to \times . We will discuss this second change in a moment.

An “arg max,” or maximizing argument, tells you which element(s) of some set maximizes some function when used as that function’s argument. In this case, that function is $f(x) \mapsto \text{cat}'(x) \times \text{black}'(x)$, which is a function that tells you *to what extent* something is a “black cat.” This is similar to our earlier uniqueness constraint, except more than one entity can have the maximum value at the same time. Also, an increase in either of the factors (e.g. an increase in the extent to which x is black or the extent to which x is a cat) leads to an overall increase in the product of the two. Recall that since $\text{cat}'(z)$ and $\text{black}'(z)$ are each in D_t , the product of the two, $\text{cat}'(z) \times \text{black}'(z)$, is also in D_t . So, just as D_t was closed under \wedge , we maintain that D_t is closed under \times , and what’s more, if *either* $\text{black}'(z)$ *or* $\text{cat}'(z)$ increase, their product increases as well. Likewise, if either of the things we are multiplying decreases, their product will decrease as well. This gives us a helpful heuristic, which can be stored by the semantic system / lexicon, for which candidates to prioritize evaluating first when searching for the unique entity that satisfies some predicate:

$$\begin{aligned}
& \mathbb{E}[\text{cat}'(z) \times \text{black}'(z) \mid \text{black}'(z) > \text{black}'(w)] \\
& > \\
& \mathbb{E}[\text{cat}'(w) \times \text{black}'(w) \mid \text{black}'(z) > \text{black}'(w)]
\end{aligned}$$

In other words, we're saying that we're going to prioritize evaluating the blacker entities first, since the more black an entity is, the more likely it is to be both the blackest and cattiest entity in the domain of discourse. I say more about the *expectation operator* \mathbb{E} below, but the takeaway is that by continuously choosing entities that increase along either of the 2 axes, we can guarantee that we will eventually reach an entity that best satisfies the predicate among all of its neighbors, and instead of checking every single entity in the context (or every single entity in every single sub-context), we only have to search an exponentially smaller space of entities. Limiting the exponentially large search space in this way would not be possible if truth values were only 0 or 1, since whether something definitely is or definitely is not, say, "black" cannot give you any useful information about what candidates to consider next.

The expectation operator \mathbb{E} simply tells us the expected truth value of some statement, and of course, this value can be between 0 and 1. The operator can be used with conditional statements as well. This allows us to discuss an array of semantic relationships that would be difficult to describe otherwise. For example, our semantics should be able to describe why certain lies are more believable than others:

$$\begin{aligned}
\mathbb{E}[\text{I see 2 moons in the sky.}] &\leq \mathbb{E}[\text{I see 2 moons in the sky.} \mid \text{I am on Mars.}] \\
\mathbb{E}[\text{I can fly.}] &\leq \mathbb{E}[\text{I can fly.} \mid \text{I am a pilot.}]
\end{aligned}$$

Notice that regardless of whether or not you knew that both of these statements are false, our intuition tells us that the statements are *more believable* in the case that the statement after the \mid is true. This allows us to draw in-

equalities, in addition to pure equalities and non-equalities, between the truth values of statements.

I say more about the expectation operator below, however I have not yet motivated the expansion from the binary \wedge operation to \times . This brings us to our next relation, predicate modification. Predicate modification remains

$$\llbracket X_{\langle e,t \rangle}, Y_{\langle e,t \rangle} \rrbracket \mapsto \llbracket Z_{\langle e,t \rangle} \rrbracket$$

and instead of $\llbracket Z \rrbracket$ being strictly defined as

$$\llbracket Z \rrbracket = \lambda x_e [\llbracket X \rrbracket (x) \wedge \llbracket Y \rrbracket (x)],$$

we can now use multiplication in place of \wedge to expand the definition of the denotation of the modified predicate, in place of binary *and*.

$$\llbracket Z \rrbracket = \lambda x_e [\llbracket X \rrbracket (x) \times \llbracket Y \rrbracket (x)].$$

Note that this is simply a generalization of our earlier model, since we derive the same truth values for binary predicates: for any $x \in D_t = [0, 1]$, i.e. for any truth value $0 \leq x \leq 1$,

$$0 = 0 \wedge x \Leftrightarrow x \times 0 = 0,$$

and

$$1 = 1 \wedge x \Leftrightarrow x \times 1 = 1.$$

This symmetry should be encouraging. This means that the new model already inherently includes the same predictions that the binary model made, *and* admits more fine-grained predictions on top! There is also a more general symmetry at play here: Function application, which applies functions defined by recursive *cross products* of the form

$$D_a \times D_b,$$

appears much more similar to predicate modification when it applies to func-

tions defined by recursive *multiplicative products* of truth values of the form

$$p \times q.$$

Now, in the case of predicate abstraction, our final relation, we give the denotation

$$\llbracket y \rrbracket = \lambda x. \llbracket Z \rrbracket^x$$

to y when y 's daughters are Z and a relative pronoun. The only minor update we need to make here, in addition to what we've already said about the difficulty of searching for an entity that fully satisfies some predicate, relates to how we interpret assignment variables. When we notate the denotation of α under x ,

$$\llbracket \alpha \rrbracket^x,$$

this x defines an assignment of denotations to each entity. We've assumed that these definitions are discrete and unique, meaning that all assignment variables assign one discrete denotation to each identity. Not only does this assumption require us to assume that entities are cognitively represented as discrete objects (e.g. $\{x : \llbracket x \text{ is a chair} \rrbracket\} \in D_e$, or $\llbracket \text{the color red} \rrbracket \in D_e$), this also does not help us explain why certain assignments are more ambiguous than others. Another consequence of this assumption is that for any two assignment variables x and y and some entity $z \in D_e$, the entity z is either assigned the exact same denotation under both x and y , or it is assigned two different denotations. Since the two denotations are either the same or different, for all pairs of assignment variables x, y ,

$$\mathbb{I}(\llbracket z \rrbracket^x = \llbracket z \rrbracket^y) \in \{0, 1\} = D_t,$$

with the statement evaluating to exactly 1 if they are the same, and exactly False if $\llbracket z \rrbracket^x$ and $\llbracket z \rrbracket^y$ have any difference between what they denote whatsoever. In a similar vein, whether two different entities are given the *same* denotation is also determined in part by an assignment variable. For example,

the context c might determine if two entities are coindexed:

$$\mathbb{I}(\llbracket \text{my professor} \rrbracket^c = \llbracket \text{Matt} \rrbracket^c) \in \{0, 1\} = D_t.$$

I leave it to you to conjure values for c that cause the expression above to evaluate to either 0 or to 1. However, I remind you that the only change we're proposing is an expansion from $D_t = \{0, 1\}$ to $D_{\textcolor{red}{t}} = [0, 1]$. The natural question, then, is whether we can continue to make this change consistently, and evaluate assignment variables as if they assign *degrees* of equality between entities instead of binary values. Let

- c_0 : I am in Times Square.
- c_1 : I am a mile away from Times Square.
- c_2 : I am two miles away from Times Square.
- c_3 : I am three miles away from Times Square.
- ...
- c_n : I am n miles away from Times Square.

I hope it is uncontroversial to say that

$$\llbracket \text{I like New York City.} \rrbracket^{c_0} \text{ is entailed by } \llbracket \text{I like this place.} \rrbracket^{c_0}$$

and that

$$\llbracket \text{I like New York City.} \rrbracket^{c_{1,000}} \text{ is not entailed by } \llbracket \text{I like this place.} \rrbracket^{c_{1,000}}.$$

So, either there exists an arbitrary threshold i in $0 < i < 1,000$ where this entailment suddenly stops holding, or it must be that our prior statements about entailment relationships actually exist in $D_{\textcolor{red}{t}}$, not D_t (a third option is to append new machinery to our discretized, binary system in other novel ways). If this is the case, that means that contexts assign denotations to entities like “this place” in a continuous and distributed fashion.

Another natural extension to our notation, but not to our theory, allows us to explicitly notate things previously “swept under the c ”. Often, when we write

c in a denotation’s notation, we are not referring to any specific context c , but instead we are actually trying to refer to some general, “default” context in which we would like to make formal claims. We also have the option not to write anything at all in the superscript of our double brackets, which is understood to mean the same thing as this generalized c . We can instead notate assignment variables explicitly and consistently again using the expectation operator \mathbb{E} from probability theory:

$$[\![\alpha]\!]^c = \mathbb{E}_{\sim c}([\![\alpha]\!])$$

This would be read “the denotation of α under c is the expected denotation of α sampled from the context c . This “sampling” procedure is then defined by c , and is precisely the “method of choosing (sampling) which candidates to evaluate” for which we were looking for an explanation earlier. This allows us to do away with “partial functions” and questions about the role of the parser in implementing g , since we have a concrete and plausible description of how this assignment is calculated. If the context c is actually meant to be unspecified, e.g. if we are only writing c out of habit when notating a denotation that’s unrelated to some specific context, then we can just *not write it*, and our derivation is unaffected!

$$[\![\alpha]\!] = \mathbb{E}[\![\alpha]\!] = \mathbb{E}([\![\alpha]\!]) = [\![\alpha]\!] = [\![\alpha]\!]$$

Now, is this probabilistic expansion of our semantic model valid? Does it answer questions we couldn’t answer before, or provide simpler explanations to the questions we could answer? There are in fact many benefits to this approach beyond what we have already discussed. We’ve already seen that this approach scaffolds nicely on top of our prior model, and that we can use it to precisely understand the role of c and other assignment variables up front without hand-waving, appending complex new machinery, or substituting probabilistic semantic descriptions with infeasible discretized pragmatic burdens (e.g. claiming that the brain is capable of efficiently checking every subset of some domain for whether it satisfies some condition). We can also

give a relatively simple denotation for the question at hand:

$$\llbracket \text{many NP VP} \rrbracket^c = \mathbb{E} [| \text{NP} \cap \text{VP} | > \mathbb{E} [| \text{NP} \cap \text{VP} |]]$$

Here, the right-hand-side can be read (or paraphrased) as

“The number of NPs that I expect to VP in the context c is greater than the number of NPs that I would expect one to expect to VP.”

I can imagine other areas of our current semantic model where this relatively minor expansion of our axioms from $D_t = \{0, 1\}$ to $D_t = [0, 1]$ could be fruitful. For one, by allowing truth values between 0 and 1, we can now provide precise definitions of our inference relations using further inequality relationships between the expected truth value of conditional statements. Let $x, y \in D_t$. Then,

<i>Metalinguistic statement</i>	<i>Definition</i>
“ x implicates y ”	$1 > \mathbb{E}[y x] > \mathbb{E}[y]$
“ x entails y ”	$1 = \mathbb{E}[y x] \geq \mathbb{E}[y]$
“ x presupposes y ”	$\left\{ \begin{array}{l} \mathbb{E}[x] = \mathbb{E}[x y] \times y \\ \leftrightarrow \\ \mathbb{E}[\neg x] = \mathbb{E}[\neg x y] \times y \end{array} \right.$

The two equalities in the last definition are equivalent to one another, we are simply stating the same constraint in different ways. I won’t belabor a demonstration of each, but as far as I can tell, this simple table allows us to precisely define what each of these inference relations are. I also claim without a shred of evidence that we can use this model to construct experimentally verifiable, simpler, and/or more accurate explanations of phenomena such as scalar implicature, presupposition failure, presupposition filtering, predicates like “big,” bleached conditionals, and resolving scope ambiguity.

- (d) $\llbracket \text{at least seven NP VP} \rrbracket^c = 1$ iff $|\text{NP} \cap \text{VP}| \geq 7$
- (e) $\llbracket \text{all but two NP VP} \rrbracket^c = 1$ iff $|\text{NP} \setminus \text{VP}| \leq 2$
- (f) $\llbracket \text{most NP VP} \rrbracket^c = 1$ iff $2 \cdot |\text{NP} \cap \text{VP}| > |\text{NP}| + n$.

Alternatively, instead of describing “most” by using an arbitrary value $n > 0$ that is the same across every person’s semantic system, we can capture our intuition that this “most” is *increasingly* true as we move past half by writing that

$$\llbracket \text{most NP VP} \rrbracket^c = \mathbb{E}_{x \sim c(D_e)}[\text{VP}'(x) > 0.5 | \text{NP}'(x)].$$