

Models for the Semantic Interpretation of Structured Propositions and Semantic Representations

David Goss-Grubbs
davidgg@u.washington.edu

November 19, 2006

1 Introduction

This paper briefly reviews an approach to the semantics of propositions whereby the semantic value of a natural language sentence is a *structured proposition* (King 2006). It is then argued that structured propositions are not, strictly speaking, semantic objects, but rather formal objects, which require interpretation by some system of semantic rules in order to determine the conditions under which they are true.

Structured propositions are compared with the sorts of semantic representations found in computational applications, and are judged to be similar in essential ways. It is claimed that any model-theoretic semantics capable of interpreting structured propositions will also suffice, at least in its broad outlines, for interpreting the sorts of semantic representations used in computational applications.

Different systems of structured propositions or semantic interpretations will require different rules for their model-theoretic semantics. This paper describes a class of model in terms of which any such set of semantic rules can specify truth conditions. One nice thing about these models is that they are entirely “flat” first-order models, and can be implemented quite directly on standard databases.

2 Structured Propositions

Frege’s (1892) distinction between the sense and the reference of an expression explains why (1) is more informative than (2), even though *the morning*

star and *the evening star* have the same reference: the planet Venus. Similarly, this distinction shows us why (3) and (4) are not synonymous even if the reference of *Sandy is tall* and *Sandy is asleep* is the same (e.g. they are both true).

- (1) The morning star is the evening star.
- (2) The morning star is the morning star.
- (3) Kim believes Sandy is tall.
- (4) Kim believes Sandy is asleep.

In both of these cases, it is the sense of these expressions which distinguish the meanings of the sentences, not their reference.

One standard definition of the sense of an expression is the expression's intension (Carnap 1947), which is a function from possible worlds to the expression's extension. Since *the morning star* won't have the same extension as *the evening star* in every possible world, and *Sandy is tall* and *Sandy is asleep* won't be true in all the same possible worlds, this seems to solve the problem.

However, intensions are widely understood still to be inadequate. (See, e.g. Soames 1987.) One problem is that there are some propositions that are true in all the same worlds, which nonetheless must be semantically distinguished in some contexts.

The object of *believe* in each of sentences (5) and (6) is a proposition which is true in every possible world. They must therefore be, according to the intension approach, the same proposition. Yet (5) and (6) fail to be synonymous.

- (5) Kim believes every bachelor is male.
- (6) Kim believes every prime number greater than 2 is odd.

There must be something more to senses than just intensions.

A popular alternative to the intensional approach is the structured proposition approach. Here, a proposition is not just a single set-theoretic object (i.e. a function from worlds to truth values). Rather, it is an n -tuple containing a number of set-theoretic objects. For example, (7a) might be given the semantic value in (7b).

- (7) a. Kim sneezes.

b. $\langle \text{Kim}, \lambda(x).sneeze'(x) \rangle$

The sentence would be true just in case the function in the second position of the pair, when applied to the argument in the first position, yields a true statement. However it is that sentences are assigned structured propositions, it is clear that *every bachelor is male* and *every prime number greater than 2 is odd* will be assigned different ones.

But how can we go from a sentence like (7a) to a value like (7b)? In this case, the extension of the subject is placed in the first position of the proposition; and that value is also abstracted out of the extension of the sentence, and the result is placed in the second position. But this rule won't always work. By itself it can't distinguish the meanings of (8) and (9).

(8) Kim believes Felix is a black cat who is not black.

(9) Kim believes Felix is a white cat who is not white.

In each embedded sentence, the extension of the subject is Felix. The extension of the embedded sentence, with Felix abstracted away, is also the same. Whether the proposition is written as (10) or (11), it is just the function which maps every individual to *false*.

(10) $\langle \text{Felix}, \lambda x.[color'(x, \text{black}) \wedge cat'(x) \wedge \neg color'(x, \text{black})] \rangle$

(11) $\langle \text{Felix}, \lambda x.[color'(x, \text{white}) \wedge cat'(x) \wedge \neg color'(x, \text{white})] \rangle$

Cresswell and von Stechow (1982) note it is always possible to find some element to abstract out so that the right propositions are built. So (12) and (13) are different propositions from one another, in that the colors that fill their first positions are different.

(12) $\langle \text{black}, \lambda x.[color'(Felix, x) \wedge cat'(Felix) \wedge \neg color'(Felix, x)] \rangle$

(13) $\langle \text{white}, \lambda x.[color'(Felix, x) \wedge cat'(Felix) \wedge \neg color'(Felix, x)] \rangle$

But when do we use the sort of rule which would give you (10) and (11) versus the sort of rule which would give you (12) and (13)?

In Cresswell and von Stechow's system, *believe* is a three-place predicate which takes as arguments the believer, some thing, and a property the believer believes of that thing. Rather than have the semantics decide on just what the thing is (Felix vs. black) and just what the property is (being a

black cat that isn't black vs. being a color that Felix both is and isn't), all possibilities are generated. Pragmatic rules are used for determining which of these distinct semantic values to use. So the sentence (14a) gets the semantic interpretations (14b) and (14c), among others.

- (14) a. Kim believes that Fido likes Felix.
 b. $believe'(Kim, Fido, \lambda x.likes'(x, Felix))$
 c. $believe'(Kim, Felix, \lambda x.likes'(Fido, x))$

Proposition (14b) means “Kim believes of Fido that she (Fido) likes Felix”; proposition (14c) means “Kim believes of Felix that Fido likes him (Felix)”.

But we have a problem here. The truth conditions for each of these propositions had better be the same. That is, they are both true iff Kim believes that Fido likes Felix. The problem is that there is nothing in the theory that ensures that these propositions really are true in all the same circumstances. If the denotation of *believe* is just a set of triples, it is completely reasonable for the tuple $\langle Kim, Fido, liking Felix \rangle$ to be a member of that set, without $\langle Kim, Felix, being liked by Fido \rangle$ being in it. Any such condition would have to be an added stipulation, and it's not clear what such a condition could look like.

Other researchers (e.g. Soames 1987) avoid these sorts of problems by having a much more articulated structure. Sentence (15) might get a denotation like (16).

- (15) Fido likes Felix.
 (16) $\langle like', \langle Fido, Felix \rangle \rangle$

Here the first element is a predicate and the second element is the list of its arguments. In fact, such structures may be arbitrarily complex. In particular, a structured proposition will sometimes appear as an argument within another proposition. So sentence (14a) would end up with a denotation like (17).

- (17) $\langle believe', \langle Kim, \langle like', \langle Fido, Felix \rangle \rangle \rangle \rangle$

But once a structured proposition is allowed arbitrary complexity and recursion, it can no longer properly be called a semantic object. Rather it must be treated as a formal object, which requires a set of interpretation rules in order to tell whether it is true. That is to say, structured propositions, which are intended to be the semantic values of natural language

expressions, stand in need of a model-theoretic semantics of their own. This puts structured propositions in the same class as the sorts of semantic representations computational semanticists are familiar with. Both are formal systems for representing propositions, the structures that make them up and the structures they are found in. Perhaps one of the largest differences is that the basic elements of structured propositions are set-theoretic objects. The basic expressions of semantic representations often don't take that status directly (although they may be straightforwardly interpretable as such).

Due to this similarity, any model-theoretic semantics for (sufficiently complex) structured propositions would be very well-suited to semantic representations as well.

And semantic representations need a model-theoretic semantics at least as badly as structured propositions do. Just as structured propositions must be interpreted before they can really mean anything, semantic representations on their own are not particularly useful, for many applications. They must be interpreted by the application as instructions to actually do something.

This paper doesn't directly give the semantics of any particular system of structured propositions or semantic representations, as each of these would be different, depending on the system. What this paper does is specify a type of model that would support a model-theoretic semantics for any such system. Such a model needs to support everything a standard semantics supports plus the composition of propositions and their relations with other objects. The models described here do just that, and with this nice feature: they are simple first-order models, making them easy to implement in standard data stores like relational databases.

3 Models

In this section I begin describing these models by describing a rather standard, simple kind of model. I'll alter and add to that until we arrive at a system that does what we want it to do.

Typically, a model is a pair $\langle A, F \rangle$, where A is a set of individuals and F is an interpretation function. F assigns semantic values to basic expressions of the language. Typically, F assigns to predicates of the language a set of n -tuples.¹ So $F(dog) = \{x|x \text{ is a dog in the model}\}$; $F(like) = \{\langle x, y \rangle|x \text{ likes } y \text{ in the model}\}$. A model of a world where Fido is a dog, Felix is a

¹where a 1-tuple is just an individual

cat, and Felix likes Fido would look something like this:

$A =$	$\{\text{Fido, Felix, ...}\}$		
$F =$	cat	\rightarrow	$\{\text{Felix, ...}\}$
	dog	\rightarrow	$\{\text{Fido, ...}\}$
	$like$	\rightarrow	$\{\langle \text{Felix, Fido} \rangle, \dots\}$

Truth conditions within this sort of model are stated in terms of which n -tuples appear in which sets. So in (18) and (19) the sentences in (a) have the truth conditions in (b).

(18) a. Fido is a dog.

b. $\text{Fido} \in F(dog)$

(19) a. Felix likes Fido.

b. $\langle \text{Felix, Fido} \rangle \in F(like)$

One way of interpreting the n -tuples which are the output of F is that these sets and relations contain descriptions of simple propositions, where *simple* means that they are predicates whose arguments are all filled by individuals. The n -tuples that are found in the output of F are the descriptions of all and only the true simple propositions. That is, being in F is what makes a simple proposition true.

Quantified propositions don't appear directly in F . They are given slightly different sorts of truth conditions. Sentence (20), for instance, would be given truth conditions something like in (21).

(20) Every dog likes Felix.

(21) $\forall x[[x \in F(dog)] \rightarrow [\langle x, \text{Felix} \rangle \in F(like)]]$

This would be fine if the only propositions we wanted to model were the true propositions. But we also want to represent false propositions, which can play many roles within some true proposition, including occurring as the object of a belief. But in the sort of model described above, there is no way of representing a false proposition - only true propositions have any existence in the model.

4 Proposition Keys

As the first step in altering the standard model so that it supports propositions, let us add a new set P to the definition of a model. This is the set of *proposition keys*. Furthermore, let us add to every tuple in the output of F a new position, and fill it with a proposition key. These proposition keys must be unique, not only within each relation, but globally. So every tuple is distinguished from every other tuple by a unique proposition key. Finally, let all possible tuples appear in each relation. The idea is to directly model every possible simple proposition, and mark it with a proposition key. Proposition keys may be understood in much the same spirit as handles in Minimal Recursion Semantics (MRS) (Copestake et al, 2005), with the following two exceptions. First, handles are parts of MRS expressions, while proposition keys are elements in a model. Second, in MRS, two elementary predications may have the same handle as their labels; in the models described here, each simple proposition must have a unique proposition key.

For example, $F(dog)$ will no longer contain just a set whose members are the actual dogs. Rather, it will contain an ordered pair for each individual in A . The first element in that pair will be a proposition key, and the second element will be the individual. So the tuple $\langle p_1, \text{Fido} \rangle$, occurring in the relation $F(dog)$, will be the model of the proposition that Fido is a dog, labeled with the proposition key p_1 . Similarly $\langle p_2, \text{Felix} \rangle$ occurring in the relation $F(dog)$ will be the model of the proposition that Felix is a dog. In this way we model all the (simple) propositions, both true and false. In order to distinguish the true propositions from the false ones, we add another new set, R , a subset of the proposition labels, to the definition of a model. R contains the proposition keys of just the true propositions. So now, being in F isn't what makes a proposition true; it is the proposition's key appearing in R that makes it true.

So in this system, a model of a world where Fido is a dog, Felix is a cat, and Felix likes Fido would look like this:

$A =$	$\{\text{Fido, Felix, } \dots\}$
$P =$	$\{p_1, p_2, \dots\}$
$F =$	$cat \rightarrow \{\langle p_1, \text{Fido} \rangle, \langle p_2, \text{Felix} \rangle, \dots\}$ $dog \rightarrow \{\langle p_3, \text{Fido} \rangle, \langle p_4, \text{Felix} \rangle, \dots\}$ $like \rightarrow \{\langle p_5, \text{Felix, Fido} \rangle, \langle p_6, \text{Fido, Felix} \rangle, \dots\}$
$R =$	$\{p_2, p_3, p_5, \dots\}$

Truth conditions in this sort of model are stated in terms of which propo-

sition keys appear in R . So the truth conditions of sentences (18a) and (19a) would be the conditions given in (22) and (23).²

$$(22) \mathcal{I}[\langle p, \text{Fido} \rangle \in F(\text{dog})] \in R$$

$$(23) \mathcal{I}[\langle p, \text{Felix}, \text{Fido} \rangle \in F(\text{like})] \in R$$

The truth conditions for quantified sentences are altered in a similar way. The truth conditions for (20) are shown in (24).

$$(24) \forall x[(\mathcal{I}[\langle p, x \rangle \in F(\text{dog})] \in R) \rightarrow (\mathcal{I}[\langle p, x, \text{Felix} \rangle \in F(\text{like})] \in R)]$$

This system can do whatever the previous, simpler system can do. The advantage is that it can also support propositional attitude reports, making all the distinctions that the structured propositions approach can make and that cannot be made by the standard intension approach. This is because proposition keys can appear as arguments (not just labels) of certain predicates, such as *believe*. So a model that includes the information that Felix believes Fido barks would look like this:

$A =$	$\{\text{Fido}, \text{Felix}, \dots\}$
$P =$	$\{p_1, p_2, \dots\}$
$F =$	$\begin{array}{ll} \text{cat} & \rightarrow \{\langle p_1, \text{Fido} \rangle, \langle p_2, \text{Felix} \rangle, \dots\} \\ \text{dog} & \rightarrow \{\langle p_3, \text{Fido} \rangle, \langle p_4, \text{Felix} \rangle, \dots\} \\ \text{bark} & \rightarrow \{\langle p_5, \text{Fido} \rangle, \langle p_6, \text{Felix} \rangle, \dots\} \\ \text{believe} & \rightarrow \{\langle p_7, \text{Felix}, p_5 \rangle, \dots\} \end{array}$
$R =$	$\{p_2, p_3, p_7, \dots\}$

Here, *believe* is taken to be a two place relation between individuals and propositions. Proposition p_5 is the proposition that Fido barks. Proposition p_7 is the proposition that Felix believes proposition p_5 . That is, that Felix believes Fido barks. The proposition key p_7 appears in R , indicating that Felix does indeed believe that Fido barks. Note that this is entirely independent from whether p_5 appears in R ; that is, whether Fido actually barks. So the truth conditions for (25) would be as shown in (26).

(25) Felix believes that Fido barks.

$$(26) \mathcal{I}_x[\langle p_x, \text{Felix}, \mathcal{I}_y[\langle p_y, \text{Fido} \rangle \in F(\text{bark})] \rangle \in F(\text{believe})] \in R$$

²The notation $\mathcal{I}[\dots]$ indicates the (single) p such that \dots is true.

5 Embedded Quantifiers

Truth condition (24) is an example with quantification. Truth condition (26) is an example with an embedded proposition. What about sentences with embedded, quantified propositions? Let's look at sentence (27).

(27) Felix believes every dog barks.

This sentence is ambiguous. What we have seen so far lets us state truth conditions for the reading where *every dog* takes wide scope over *believes*. They are shown in (28).

(28) $\forall x[[\eta p[\langle p, x \rangle \in F(dog)] \in R] \rightarrow [\eta p_z[\langle p_z, \text{Felix}, \eta p_y[\langle p_y, x \rangle \in F(bark)] \in F(believe)] \in R]]$

This condition says that for every individual that actually is a dog, Felix believes that that individual barks. But there is also a reading where Felix doesn't have particular beliefs about any particular individuals; Felix just believes that all the dogs, whoever they may be, bark. What we want is something like (29).

(29) $\eta p[\langle p, \text{Fido, the proposition key for every dog barks} \rangle \in F(believe)] \in R$

That is, we want to have proposition keys not just for simple propositions, but for quantificational propositions as well. *Every dog barks* is such a proposition. We want its key to appear in a tuple together with Felix, to indicate (the proposition) that Felix believes the proposition that every dog barks.

6 Kinds

In order to describe what needs to be done to support embedded quantified propositions, let us first introduce a new set K to the definition of a model. This is the set of *kinds*. Formally, a kind is like a set of propositions, each with one missing argument. You can think of a kind as a class of individuals or a property. The kinds in a model might include *dogs* and *things that bark*. Here is a sample model that contains kinds for *dogs* and *cats that like Fido*.

$A =$	$\{\text{Fido, Felix, ...}\}$
$P =$	$\{p_1, p_2, \dots\}$
$K =$	$\{k_1, k_2, \dots\}$
$F =$	$cat \rightarrow \{\langle p_1, \text{Fido} \rangle, \langle p_2, \text{Felix} \rangle, \langle k_2, \Delta \rangle, \dots\}$ $dog \rightarrow \{\langle p_3, \text{Fido} \rangle, \langle p_4, \text{Felix} \rangle, \langle k_1, \Delta \rangle, \dots\}$ $like \rightarrow \{\langle p_5, \text{Felix, Fido} \rangle, \langle k_2, \Delta, \text{Fido} \rangle, \dots\}$
$R =$	$\{\dots\}$

Here we see that, along with proposition keys, kinds are the sort of thing that can act as a label for tuples in the model. The difference with tuples that are labeled with kinds is that one slot of each is filled by the special value Δ rather than by an individual. This is the slot over which the tuple is abstracted.

In the above model, the kind *dogs* is kind k_1 . It is associated with one tuple of $F(dog)$, where its only argument has the value Δ . That is, in order for something to be an instance of the kind *dogs*, it would have to be able to replace the Δ to yield a true proposition. The kind k_2 is associated with a tuple of $F(cat)$, indicating k_2 is a kind of cat, and a tuple of $F(like)$, indicating k_2 is a kind of thing that likes Fido. Together, this means that k_2 is the kind *cats that like Fido*.

So how do kinds help us with embedded, quantified propositions, such as the one in (27)? The solution begins by thinking of quantifiers as relations between kinds. A sentence containing a quantifier, then, can be seen as a proposition whose relation is the interpretation of the quantifier. So (30) says that the kind *dogs* stands in the *every* relation with the kind *barkers*.

(30) Every dog barks.

Every is a relation between two kinds. It means that everything which is an instance of the first kind is also an instance of the second. The only thing left is to add relations like *every* to our model. Here is a model in which (27) (where *every dog* has narrow scope) is true:

$A =$	$\{\text{Fido, Felix, ...}\}$
$P =$	$\{p_1, p_2, \dots\}$
$K =$	$\{k_1, k_2, \dots\}$
$F =$	$dog \rightarrow \{\langle p_1, \text{Fido} \rangle, \langle p_2, \text{Felix} \rangle, \langle k_1, \Delta \rangle, \dots\}$ $bark \rightarrow \{\langle p_3, \text{Fido} \rangle, \langle p_4, \text{Felix} \rangle, \langle k_2, \Delta \rangle, \dots\}$ $every \rightarrow \{\langle p_5, k_1, k_2 \rangle, \dots\}$ $believe \rightarrow \{\langle p_6, \text{Felix, } p_5 \rangle, \dots\}$
$R =$	$\{p_6, \dots\}$

The kind k_1 is dogs. The kind k_2 is things that bark. Proposition p_5 is the proposition that every dog barks. Proposition p_6 is the (true) proposition that Felix believes p_5 . The truth conditions for (27) in this sort of model would be as in (31).

$$(31) \quad \eta_z[\langle p_z, \text{Felix}, \eta_y[\langle p_y, \text{Min}(k)[\langle k, \Delta \rangle \in F(\text{dog})], \\ \text{Min}(k)[\langle k, \Delta \rangle \in F(\text{bark})] \rangle \in F(\text{every})] \rangle \in F(\text{believe})] \in R$$

The notation $\text{Min}(k)[\dots]$ is used to refer to the “minimal” kind k such that the material within the brackets is true, and is the one that is associated with the fewest tuples. So if kind x is *black cats* and kind y is *black cats that like Felix*, then the value of (32) will be kind x rather than kind y , because x is defined by a two tuples, and y is defined by three.

$$(32) \quad \text{Min}(k)[\langle k, \Delta \rangle \in F(\text{black}) \wedge \langle k, \Delta \rangle \in F(\text{cat})]$$

Now that we have relations like *every* in our model, there appear to be two distinct ways of determining the truth conditions of a quantified sentence. It looks like sentence (30) can have either (33) or (34) as its truth conditions. We can think of (33) as the rule-oriented truth conditions, and (34) as the fact-oriented truth conditions.

$$(33) \quad \eta[\langle p, \text{Min}(k)[\langle k, \Delta \rangle \in F(\text{dog})], \text{Min}(k)[\langle k, \Delta \rangle \in F(\text{bark})] \rangle \\ \in F(\text{every})] \in R$$

$$(34) \quad \forall x[(\eta[\langle p, x \rangle \in F(\text{dog})] \in R) \rightarrow (\eta[\langle p, x \rangle \in F(\text{bark})] \in R)]$$

This looks like a bit of a problem, because there doesn’t appear to be anything that prevents these two expressions from having different truth values. The proposition key for *every dog barks* may be in R at the same time there is an individual whose ‘dog’ tuple is in R , but whose ‘bark’ tuple is not.

The position taken in this paper is that it is up to the semantic rules of the particular system of structured propositions or semantic representations to choose whether and when to use either or both of these methods. One workable approach would be always to use fact-oriented truth conditions for matrix sentences. (Using rule-oriented truth conditions is the only possibility for embedded sentences.) On the practical side, it could be useful to have the truth conditions of (30) be the disjunction of (33) and (34). This would give us the ability to store just the rule that every dog barks without having to store barking facts about every individual dog. But this approach requires some way of keeping the facts and the rules consistent with each other.

7 Conclusion

This paper has drawn a parallel between structured propositions and semantic representations, noting that both require a system of semantic rules in order to interpret them model-theoretically. A class of semantic model was then introduced and described that is capable of supporting such systems of semantic rules. These models are flat, in the sense that they are made up of nothing more than sets of atomic objects or relations between atomic objects. This makes them relatively straightforward to implement computationally.

These models store propositions directly, giving each one a key by which it can be referenced elsewhere in the model. The model distinguishes true propositions by keeping a record of their keys in a special set. The model relates propositions (whether or not they are true) to other objects by relating their keys to those objects. Propositions containing quantified arguments are understood to be (tuples in) relations between kinds, where a kind is equivalent to a property, which is defined by a set of abstracted propositions.

References

- [1] Carnap, Rudolf, 1947 *Meaning and Necessity*. Chicago, University of Chicago Press.
- [2] Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005, 'Minimal Recursion Semantics: An Introduction' *Research on Language and Computation* 3, 281-332.
- [3] Cresswell, M. J., and A. von Stechow, 1982, 'De re belief generalized' *Linguistics and Philosophy* 5, 503-535.
- [4] Frege, Gottlob, 1892, 'On Sense and Reference', in *Translations from the Philosophical Writings of Gottlob Frege*, Geach and Black (eds.), Basil Blackwell, Oxford, 1977.
- [5] King, Jeffrey C., "Structured Propositions", *The Stanford Encyclopedia of Philosophy (Fall 2006 Edition)*, Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/fall2006/entries/propositions-structured/>.
- [6] Soames, Scott, 1987 'Direct Reference, Propositional Attitudes and Semantic Content', *Philosophical Topics* 15, 47-87.