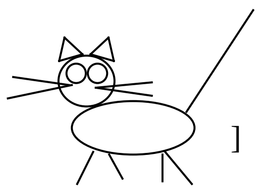
Answers to the requested questions from homework #9:
(Questions in italics, answers in regular face)

Katie: *Why isn't it a problem when we put the word itself in its own lexical entry, e.g. [[cat]] = [$\lambda x \in D_e$. x is a cat]?*

Erika: *I expect that this question is more an issue for next semester's class, but it's been bugging me all the same. It looks like the kind of interpretation we're making of sentences is very restricted by rules, to the point where I expect a computer could easily 'compute' the denotation of a given sentence if provided with all the appropriate rules and lexical entries. I like this; it means that at the very least we're being consistent in how we go about applying rules and computing denotations. Where a computer would fail, I expect, is in the assignment of lexical entries. It feels a little like we're cheating when we say, for instance, [[cat]] = [x . x is a cat], because the definition feels so circular. We can't define "cat" without making reference to cats. In a similar way, the human language speaker doing the computation of the denotation of a sentence has to be able to use his/her knowledge of the language to determine what the appropriate meaning of a word with multiple meanings (e.g., 'bank', but arguably most words) to use as the lexical entry of a word. I guess I want to see even this level as rule-guided as the rest of our theory.*
Perhaps my question is whether our level of semantics cares about how it actually refers to the world. It feels like all we're doing is playing with rules as far as we can until we get to the point where you need to have an understanding of human language and human reference (and even syntax) to understand what's left. (For instance, I'm not convinced that it's not trivial to translate [[Rover chased Dixie]] into = 1 iff Rover chased Dixie.) Is there an assumption that we're moving from some kind of surface structure to its corresponding deep structure?

Heidi: In deriving an account of the meaning of our predicates, it does certainly seem like cheating to write something like "x is a cat" is the truth-condition that applies when you're looking up the meaning of "cat". Now, there's a short and a long answer to this question.

Short one first, which is really a cop-out for what E&K are really trying to get at:
The truth-condition "x-is-a-cat" is just shorthand for the real-world situation in which the referent of "x" is a cat. That is, "x-is-a-cat" isn't really English; as I discussed very early in the course, it's a meta-language we're using to represent concepts _directly_. If we knew how to draw a picture of the real-world situation of being a cat, that would do just as well for a representation of the truth condition, something like this, perhaps:

[(x D_e . x = ]

So in a very real way, we _don't_ actually have the word "cat" in our definition of [[cat]] -- we have a shorthand for representation of a real-world situation, in which cattiness is a property we're ascribing to something. Similarly for "gray" and "happy" and "small" and everything else.(Cash prize: draw a convincing ASCII picture of happiness (not a smiley face!! what about a "happy song?")). It's hence not trivial to compute truth conditions, because what we're doing is giving an explict account of how the concepts that are represented by the words combine to provide an interpretable sentence -- that is, how we get from the individual concepts [[chase]], [[cat]], [[dog]] and [[the]] to the very different concept [[the cat chased the dog]] -- a non-trivial enterprise. -- However, that's not really the question. The question really is, so, how do we associate lexical items with our concepts (things like the picture of cattiness I drew above)? And is this association describable in formal terms, or at least some parts of it describable in formal terms? Basically, aren't we leaving a heck of a lot out by not finding out how the lexical-item/concept relation is represented? The (direction to take to get to the beginning of the) long answer: is the following:

(a) are there restrictions on the kind of concepts we can associate with lexical items? I.e. are there concepts that are just not nameable in natural language? (The paper by Keenan and Stavi suggests an answer to this question (yes, there are unnameable concepts) in the realm of quantifiers. Note, of course, that we can _entertain_ unnameable concepts -- that is ,we can conceive of them, it's just that natural language is constructed in such a way that we can't name them; we have to refer to them compositionally).

(b) how do kids associate concepts with lexical items in such a way that their associations match the associations of all the adult speakers of the language around them?

(c) If some lexical items break down into pieces ("trans+mit+ion", "de+crease", "able+ity+s"...) do we represent the pieces separately and put them together with our language faculty in the same way we put sentences together? or do we represent them as whole items? (d)... thru (z) and beyond. Many of these questions in the long answer will turn out not to be questions about the language faculty per se, but rather about our non-linguistic conceptions. It's the hope of the formal-semantics enterprise, e.g., that the truth-condition on [[cat]] is exactly the concept of cattiness, in the non-linguistic part of our brains -- that is, that the lexical entry, as represented above, is exactly right -- it's just an instruction for where to go from the lexical item to get a particular concept, and the composition rules (FA, PM, etc.) tell you to insert it in the bigger concept you're building with the sentence in a particular way. The question K&E are asking is really, what does this instruction "really" look like, and does it have analysable component parts? For further investigation of these questions, may I recommend (plug, plug) my course on lexical semantics next term: 522? :)

Gondy: *Is it possible to make rules that define when something has a certain type (<e,t>, <e>, etc.), when certain rules need to be applied (PM, PA, etc>) in such a way that they work for all cases? In other words, has this been formalized yet in such a way that a computer could do this for us?*

Heidi: The short answer to this question is, yep, it's certainly possible. The fully-specified nature of the rules we've constructed so far will provide an interpretation for any syntactic structure whose lexical items are of appropriate types so that our particular rules can combine them. And our rules are fully specified for the environments in which they can apply, so there wouldn't be any problem with using them as interpretation instructions in a parsing program, say.

However, as for applying in _all_ cases, we're basically only at the beginning. So far, for instance, we can't interpret tense markers or modality markers ("Felix _can_ chase Rover"). We don't have lexical items for verbs which take a clausal complement ("John _thinks_ that Sue left"). We haven't represented any adverbs ("Felix often chases Rover"). So our particular fragment of English is a teeny fragment indeed.

That's not to say that a much more comprehensive and adequate treatment of these types of phenomena don't exist in a framework like ours; there's lots of smart semanticists out there chugging away dealing with these and other questions. And assuming that they're not stepping out of the realm of the fully formally specified, it should in principle be possible to use their analyses to build a good parser. Certainly some work like this has been done (I can try and find some good references if you're interested). BUT when you're actually trying to get the computer to *understand* the truth-condition that it's computed, you're into the problem of giving the computer a conceptual representation of the world -- the problem of how to give the computer lexical entries that capture all the information about cattiness that we're capturing by saying "x-is-a-cat". And then we're back to the problem that Erika and Katie are asking about. So a lot of the work on natural language processing, as you no doubt know, is concerned with lexical semantics. Things like WordNet, that enormous collection of entailment relations and meaning networks among words, is an attempt to model conceptual knowledge for a computer's use. It might also be a theoretical approach to representing the way _we_ represent and store conceptual knowledge (although I think it's probably far from adequate for that task). Back to Erika's discussion: when we're trying to decide which concept represented by the lexical entry [[bank]] is the appropriate one, we consult all kinds of information about the discourse context, the speaker's intention, etc., that is extremely complicated and difficult to model -- so that's a big job for anyone trying to make a computer parse strings of English.

So, again, a short answer and a long answer -- yes it's possible (not that all the problems with the semantic approaches to various phenomena of natural language have been fully worked out); a computer could do the computations that we've been doing given a system modelled on the system we've been using. but as for making an adequate understanding of what the computer has done by doing these calculations (e.g. so it could respond to a question appropriately), that's another whole ball of wax.

Travis: *What's the denotation of "a"?*

Heidi: In the homework, and as we've been treating it so far, it's simply zero, nada, nothing, non-interpretable. So if you didn't include it in your semantic interpretations at all (i.e. just say that the denotation of the NP [a cat] is the same as the denotation of the N' and N [cat] because the semantic component simply can't _see_ it, that was an appropriate avenue to take. If the semantic component can't _see_ "a" (or, presumably whatever category label dominates it, let's say "Det"), then it'll interpret the NP [a cat] by NN -- equating it to the denotation of [cat].

Another possible approach, suggested by H&K in their short discussion of the topic at the beginning of chapter 4, is to imagine that things like "a" and "is" are simply items that take functions as arguments and give as values those very same functions -- [$\lambda f \quad D_{<e,t>}$. f]. If that is the meaning you assume for "a", then when the semantic component gets to the NP, it'll interpret it by FA, rather than NN, and the denotation for [a cat] will be simply [[a]]([[cat]]), which by the definition of lambda-notation will be [[cat]].

This is the only residual point in the homework where there was a small amount of confusion. Some folks tried to treat [a cat] by functional application, so they gave as its meaning [[a]]([[cat]]), but they _also_ assumed that "a" was vacuous. So when they got to interpreting [[a]]([[cat]]), they simply said, oh, "a" is vacuous, so therefore the denotation of [[a]]([[cat]]) is equal to [[cat]]. But that can't be right; if those two are combined via F.A., then if one thing isn't an appropriate argument for the other thing, the node simply ends up without a denotation (that's why something like "John mary" doesn't mean anything). So you have to pick one approach or the other: either assume the semantics doesn't see "a" and its mother node Det, or assign it the meaning [Lf E D<e,t> .f] -- but don't try to mix 'em both. So, those were some cautionary notes about what to do with "a" when the NP is in predicate position (i.e. in a small clause or following "is"). The more astute among you will no doubt have noticed that "a" shows up in other positions -- argument positions -- as well, e.g. "A man arrived." or "Susan wants to catch a fish" or even (although we won't treat these in this course) "A dog eats meat." For the first two, although not the third, we'll be treating "a" as a quantifier, roughly equal in meaning to "one" (or "some" when it's combined with a singular N, e.g. "Some linguist arrived"). We can see that it's ambiguous because it interacts with other quantifiers to give quantifier scope ambiguities: "a man loves every woman" can mean either there's a specific man such that that man loves every woman, or that for every woman, there's a man (not necessarily the same in each case) who loves her.So that's the ambiguity associted with "a" that we're going to be dealing with. The "a dog eats meat" cases are tied up with a whole other problem, that of generics, and we're not going to get into it.

Bob: *I wonder about the full compatibility of the lambda model with syntactic theory. Would it be a fair estimation that we're creating denotation structures of PF? Would a more compatible interpretation actually create formulas out of LF? Would this require a lot more use of Predicate Abstraction, given post-spellout movement?*

(I know that's more than one question. Here's another...) If we began to include functional projections in our syntactic trees, I suppose the formula would be a lot more complicated. Would we also be forced, basically, to give things like Infl lexical entries? This implies to me, then, that all heads are, in some respect, lexical.

Heidi: First set of questions: short answers: no, yes and yes. Actually, we'll see a bunch of how this works in the next couple of classes. Early on I alluded to Quantifier Raising as a mechanism for capturing scope ambiguities: that will be exactly an LF movement which occurs to provide an unambiguous interpretation of a syntactic tree. In general, although I haven't been explicit about this, we're always assuming that our semantic component is making formulas out of LF representations, not out of PF ones. And in fact, every time we raise a quantifier (or do any kind of A-bar movement, including question-formation), we'll be employing predicate abstraction, with various caveats and restrictions.

Second set of questions: Yep, as soon as we start trying to represent, e.g., Tense (including modals like "can" "will" and "should"), things will indeed get a lot more complicated. We certainly will need to give some Infl-like things lexical entries; there are interactions of negation and modality, and tense and NP-interpretation, and any number of other things we haven't looked at yet. Other Infl-like things won't need any semantic lexical entries: Agr, for instance, if Chomsky's right, is semantically invisible (just as we've been assuming for predicate "is" and "a" above). Syntax may impose constraints and hence motivate the existence of functional projections that have nothing to do with any semantic interpretation (although Chomsky argues that this is undesirable in a Minimalist framework). Certainly, though, anything that has semantic content will need a lexical entry, no matter whether it is word-sized or affix-sized.