# Semantics and Disambiguation

## 1 Semantics

As we have said previously, current theories of semantics assume that semantics is computed *compositionally*: the meaning of an expression is a function of the meaning of its parts. The way we achieve this is with what's called the *rule-to-rule hypothesis*: we associate a semantic rule with each syntax rule.

To give an example of this, we have to introduce some new notation: *lambda-expressions*. (Even so, our treatment is informal. A proper treatment would involve a consideration of what is called *type theory*.)

- If $X$ is a variable and $E$ is an expression, then $\lambda X[E]$ is a lambda-expression. For example, $\lambda x[\mathit{died}(x)]$ is a lambda-expression. Lambda-expressions are a way of writing functions without giving them names. Effectively, the lambda variables are the parameters, so $\lambda x[\mathit{died}(x)]$ is a function that takes in one argument.

Given that lambda-expressions denote functions, then we can write expressions where we apply the function to an actual argument. This is called *lambda-reduction*.

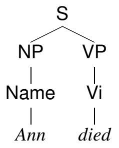
- If $\lambda X[E]$ is a lambda-expression, then $\lambda X[E](A)$ is the application of the lambda-expression to argument $A$. An example is $\lambda x[\mathit{died}(x)](\mathit{ann})$. These kinds of expressions can be simplified. You simply write the expression $E$ after replacing all occurrences of the variable $X$ in the expression $E$ by the argument $A$. For example, $\lambda x[\mathit{died}(x)](\mathit{ann})$ simplifies to *died*(*ann*).

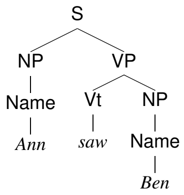
Using this notation, we can give a version of our grammar in which grammar rules are associated with their corresponding semantic rules. I'm also using some more notation. When I write, e.g., S', this means the semantics of the S.

| Grammar rule | Semantic rule |
|---|---|
| S → NP VP | S' = VP'(NP') |
| NP → Name | NP' = Name' |
| VP → Vi | VP' = Vi' |
| VP → Vt NP | VP' = Vt'(NP') |

| Word | Category | Semantics |
|---|---|---|
| *Ann*: | Name | : *ann* |
| *Ben*: | Name | : *ben* |
| *died*: | Vi | : $\lambda x[\mathit{died}(x)]$ |
| *saw*: | Vt | : $\lambda y[\lambda x[\mathit{saw}(x,y)]]$ |

Here are parse trees for *"Ann died"* and *"Ann saw Ben"*. In the lecture, we'll compute the semantics of each node using the rules above.



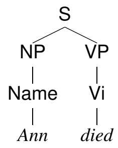

In fact, the above is a carefully chosen grammar: one with quite simple semantics. Let's also look at a slightly more complicated but slightly more realistic example:

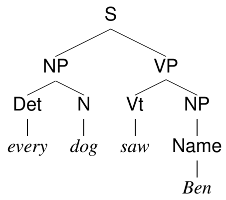


| Grammar rule | Semantic rule |
|---|---|
| S → NP VP | S' = VP'(NP') |
| NP → Name | NP' = Name' |
| NP → Det N | NP' = Det'(N') |
| VP → Vi | VP' = Vi' |
| VP → Vt NP | VP' = Vt'(NP') |

| Word | Category | Semantics |
|---|---|---|
| *Ann*: | Name | : $\lambda P[P(\mathit{ann})]$ |
| *Ben*: | Name | : $\lambda P[P(\mathit{ben})]$ |
| *every*: | Det | : $\lambda Q[\lambda P[\forall x(Q(x) \Rightarrow P(x))]]$ |
| *died*: | Vi | : $\lambda P[P(\lambda x[\mathit{died}(x)])]$ |
| *saw*: | Vt | : $\lambda P[\lambda Q[Q(\lambda x[P(\lambda y[\mathit{saw}(x,y)])])]]$ |
| *dog*: | N | : $\lambda x[\mathit{dog}(x)]$ |

Here are parse trees for *"Ann died"* and *"Every dog saw Ben"*. In the lecture, we'll compute the semantics of each node using the rules above.




Where do semantic rules come from? They come from knowledge engineers. Machine learning has made no impact here.

We are not going to give any treatment of other aspects of meaning. At this point in the development of NLP, while some of the work on, e.g., pragmatics is very sophisticated, taken as a whole it is still something of a rag-bag. The only observation we will make is: pragmatics is not compositional, as can be seen from the following examples:

(1) *"Ben loves himself."*

(2) *"Ann loves Ben's dog. It barks with pleasure when it sees her.'*

(3) *"Ann kicked the bucket."*

## 2 Disambiguation

The perspective we will take on ambiguity and disambiguation might be called a *generate-and-test* approach. For an ambiguous string, multiple wffs are generated and are then tested for plausibility by some other process. In the ideal, this other process rejects all but the speaker's intended meaning of the input string.

However, it is by no means certain that generate-and-test is the right approach. Quite simple sentences can have hundreds, and even thousands, of readings, and thus demand too much space and time for the approach to be practical. Furthermore, humans rarely perceive ambiguities, which suggests that approaches in which large numbers of readings are generated are psychologically implausible.

Despite its implausibility, for the rest of this section we assume the generate-and-test approach. We do this because it is simple. Let's look at several ways of choosing between different readings of a sentence.

**Probabilities.** In probabilistic context-free phrase-structure grammars, each grammar rule has a probability associated with it. The sum of the probabilities for all rules with the same left-hand side is 1. The probability of any particular tree is, e.g., the product of the probabilities of all the rules used in the tree.

The probabilities for the rules can be learned from a body of manually parsed text.

Probabilities can also be placed on the different meanings of each word. For example, *"pen"* is more frequently a writing implement, less frequently an enclosure and least frequently a female swan. Probabilities can be chosen to reflect this.

Ideally, these probabilities should be context-sensitive (conditional probabilities). In a discussion of farming, the very presence of words such as *"sheep"* in the nearby text should increase the probability that *"pen"* means an enclosure. However, even context-sensitive probabilities will not do a perfect job:

(4) *"His pen flew across the page."*

The presence of *"flew"* increases the probability that *"pen"* means a female swan. But, most of us would interpret this as speedy writing, so *"pen"* is a writing implement.

**Sortal restrictions.** We can group individuals into *classes*. For example, there are individuals that are physical objects and those that are abstract objects. And we can devise a *class hierarchy*. For example, the physical objects subdivide into the animate ones and inanimate ones; the abstract objects subdivide into the events, processes and states. These subclasses might further subdivide into their own subclasses.

It is clear that the arguments of a relation or a function are often restricted to individuals of certain classes. For example, the two-place 'drink' relation requires the drinker to be animate and the object drunk to be a fluid physical object; the one-place 'crawl' relation requires the crawler to be an animal. These restrictions on the sorts of the arguments of relations are referred to as *sortal restrictions* or *selection restrictions*.

Wffs that would violate sortal restrictions can be rejected and alternative, sortal restriction-preserving readings sought. This might disambiguate an ambiguous utterance. For example,

(5) *"A bug crawled into the sugar bowl."*

contains an ambiguity: the word *"bug"* may denote insects or microphones. However, the sortal restriction on the argument of *"crawled"* dictates that only the former meaning gives rise to a meaning for the sentence as a whole.

On the other hand, it might reveal what one would regard as non-literal uses of language. In sentence (6), for example,

(6) *"My car drinks petrol."*

the literal reading would violate a sortal restriction, but a non-literal reading might be possible.

One major criticism of the use of sortal restrictions is that their enforcement is inflexible. If we make the constraints very specific (as may be needed for high discriminating power), they may reject perfectly acceptable utterances. But as we 'loosen' them to accommodate a wider range of uses of a verb, we may end up with vacuous constraints that are rarely violated. For example, what is the sortal restriction on the killer in the 'kill' relation? Human murderers, wild animals, missiles, diseases, and bad news: all can be killers. Similarly, what is the sortal restriction on the killed object in the 'kill' relation? Living things, time, proposals and interest can all be killed. The sortal restrictions would appear to be that the killer must be a thing, and the killed object must be a thing. This is no constraint at all! This has led to the proposal that these sortal restrictions be treated as *preferences* rather than as absolute restrictions. Violation of a sortal preference would not cause a sentence reading to be rejected. Instead, by keeping some measure of the number and severity of the violation in each reading, readings can be ranked.

**The back-end system.** If you have computed several wffs, you can always see which makes most sense in the back-end system. For example, if you are building a natural language interface to a database system, you can see which wff ties in best with the tables in the database. If the database is about students, their modules and their grades, then *"course"* is likely to be a unit of teaching and not a place for racing or a serving of a meal.

**Ask the user.** As a last resort, the user can be consulted and asked to choose between wffs. The problem with this is that users who have already had to re-phrase and re-type their input one or more times due to the fragility of the grammar and lexicon may be unimpressed by then being asked to choose a reading.



Incidentally, note that there are, of course, some utterances, such as jokes, where the whole point is that disambiguation should fail (or, at least, the first meaning that gets chosen should be an implausible one), e.g.:

(7) *"She criticised his apartment so he knocked her flat."*

## Exercise

1. What is meant by *compositional semantics*?
2. What is meant in compositional semantics by *the rule-to-rule hypothesis*?
3. Here is a Context-Free Phrase-Structure Grammar and lexicon for a fragment of the English language in which grammar rules and lexical entries are paired with semantic rules (using the notation from the lectures):

| Grammar rule | Semantic rule | Word | Category | Semantics |
|---|---|---|---|---|
| S → NP VP | S′ = VP′(NP′) | *Becks* | Name | $\lambda P[P(becks)]$ |
| NP → Name | NP′ = Name′ | *Posh* | Name | $\lambda P[P(posh)]$ |
| NP → Det N | NP′ = Det′(N′) | *every* | Det | $\lambda Q[\lambda P[\forall x(Q(x) \Rightarrow P(x))]]$ |
| VP → Vi | VP′ = Vi′ | *player* | N | $\lambda x[player(x)]$ |
| VP → Vt NP | VP′ = Vt′(NP′) | *scored* | Vi | $\lambda P[P(\lambda x[scored(x)])]$ |
| | | *hates* | Vt | $\lambda P[\lambda Q[Q(\lambda x[P(\lambda y[hates(x,y)])])]]$ |

   Draw a parse tree for each of the following sentences. Annotate each node of your trees with its semantics. Use lambda-reduction to simplify as much as possible. Show your working.

   (a) *Becks scored*
   (b) *every player hates Posh*

4. Each of the following sentences is ambiguous in multiple ways. For each sentence, list the ambiguities and the type of each ambiguity (structural, lexical, etc.)

   (a) *Becks reached the ball.*
   (b) *Becks kissed a former lady friend.*
   (c) *Posh hit the photographer with his tripod.*

5. Identify the knowledge that would enable an agent to resolve the referents of the underlined pronouns (i.e. to whom they refer):

   (a) *Posh has a Yorkshire terrier, Lucy. She sings Spice Girls hits when they go for a walk.*
   (b) *Becks always takes baby Romeo to the match. He uses the opportunity to teach him to speak.*

6. For each of the following sentences, identify the problem that the sentence causes for compositional semantics.

   (a) *Every loving son adores his mother.*
   (b) *Someone is loved by everyone.*
   (c) *Fergie gave him the sack.*