Semantics and Pragmatics

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, $\lambda x[\textit{died}(x)]$ is a lambda-expression. Lambda-expressions are a way of writing functions without giving them names. Effectively, the lambda variables are the parameters, so $\lambda x[\textit{died}(x)]$ is a function that takes in one argument.

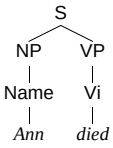
Given that lambda-expressions denote functions, then we can write expressions where we apply the function to an actual argument. This is called *lambda-reduction*.

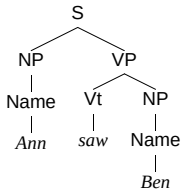
- If $\lambda X[E]$ is a lambda-expression, then $\lambda X[E](A)$ is the application of the lambda-expression to argument A. An example is $\lambda x[\textit{died}(x)](\textit{ann})$. These kinds of expressions can be simplified. You simply write the expression E after replacing all occurrences of the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

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	Word	Category	Semantics
S → NP VP	S' = VP'(NP')	*Ann* :	Name	: *ann*
NP → Name	NP' = Name'	*Ben* :	Name	: *ben*
VP → Vi	VP' = Vi'	*died* :	Vi	: $\lambda x[\textit{died}(x)]$
VP → Vt NP	VP' = Vt'(NP')	*saw* :	Vt	: $\lambda y[\lambda x[\textit{saw}(x, y)]]$

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.

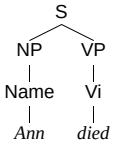


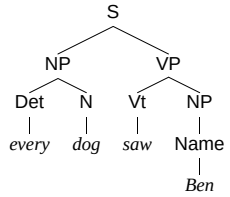


In fact, the above is a carefully chosen grammar: one with quite simple semantics. Let's also look at a slightly more complicated but slightly more realistic example:

Grammar rule	Semantic rule	Word	Category	Semantics
S → NP VP	S' = VP'(NP')	*Ann* :	Name	: $\lambda P[P(\textit{ann})]$
NP → Name	NP' = Name'	*Ben* :	Name	: $\lambda P[P(\textit{ben})]$
NP → Det N	NP' = Det'(N')	*every* :	Det	: $\lambda Q[\lambda P[\forall x(Q(x) \Rightarrow P(x))]]$
VP → Vi	VP' = Vi'	*died* :	Vi	: $\lambda P[P(\lambda x[\textit{died}(x)])]$
VP → Vt NP	VP' = Vt'(NP')	*saw* :	Vt	: $\lambda P[\lambda Q[Q(\lambda x[P(\lambda y[\textit{saw}(x, y)])])]]$
		dog :	N	: $\lambda x[\textit{dog}(x)]$

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.

2 A Few Words on Pragmatics

Recall that the concern of pragmatics is *context-dependent linguistic meaning*. It includes a rag-bag of topics: determining the referents of referring expressions (e.g. determining to whom or to what the underlined phrases refer in *"A man threw a ball to his son who caught the ball and threw it back to him."*); determining the 'force' conveyed by an utterance (e.g. working out the speech act being performed when the sentence *"Do you have change for a fiver?"* is uttered in different contexts); and handling a variety of non-literal uses of language (e.g. working out whether a sentence such as *"The kettle is boiling"* is being used literally or not).

Presently, while some of the work on pragmatics is very sophisticated, taken as a whole it is still not very successful. There are many reasons for this, including that it was initially neglected; it requires that a lot of knowledge be represented; it requires considerable reasoning capabilities; and so on. One other 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."*

The context-dependent meaning of these sentences or phrases within these sentences is not simply a function of the meaning of their parts: their context-dependent meanings arise from other parts of the sentence, previous sentences or from other knowledge entirely.

To illustrate further, we will take a simple-minded approach to the problem of reference resolution. Some NPs introduce new entities into the context; others identify entities already existing in the context. Roughly, indefinite NPs (ones where the Det is *"a"*) introduce new entities; definite NPs (ones where the Det is *"the"*) and pronouns (e.g. *"he"*, *"she"*, *"it"*) refer to existing entities. (There are numerous exceptions to the previous sentence! But it will

suffice for our purposes.) See the earlier sentence about throwing a ball: the ball is introduced by the phrase *"a ball"*; it is later referred to again by the phrases *"the ball"* and *"it"*.

The simplest way to handle reference resolution is to maintain a *history list*. This is a list of entities introduced or referred to in the text, ordered by *recency* of mention, i.e. as we search the list, we encounter the most recently-mentioned entities first. Indefinite NPs add new entities to the list. Definite NPs also add entities to the list, but the entities added are not new ones: they are found by searching through the history list. The first entity which this search encounters that is syntactically and semantically compatible with the definite NP is the referent.

Here's a simple example. We will process the following sentences:

(4) *"Ben went to see Ann yesterday. She was busy looking for Col. He…"*

We will assume that the name *"Ann"* translates to the logical constant *ann*; similarly for *"Ben"* and *"Col"*. This shows what happens:

Sentences	Actions	History list
		[ ]
Ben…	add *ben*	[*ben*]
… went to see Ann…	add *ann*	[*ann, ben*]
… yesterday. She…	search list for first female, i.e. *ann*;	
	this is the referent of *"She"*;	
	add again to the list	[*ann, ann, ben*]
… was busy looking for Col."	add *col*	[*col, ann, ann, ben*]
He…	search list for first male,	
i.e. *Col*;		
	this is the referent of *"He"*;	
	add again to the list	[*col, col, ann, ann, ben*]

The final word, *"He"*, is predicted to refer to Col. Do you agree?

In fact, this is hopelessly over-simplified but it gives the flavour. It can be made more sophisticated by making the search sensitive to more constraints and preferences in addition to recency of mention. The following are possibilities:

Frequency: Prefer more frequently mentioned entities.

(5) *"Ben was feeling good. Col went to the pub with him and Edd to celebrate. He…"*

Frequency predicts *"He"* is Ben; recency predicts Edd.

Subject NPs preferred over object NPs preferred over other NPs: This preference is based on entities that play particular roles in the previous sentence.

(6) *"Ben went to the pub with Col. He…"*

"Ben" is the subject NP in the first sentence while *"Col"* is the object NP. So, although Col is more recent, the prediction here is to prefer Ben to be the referent of *"He"*.

Structural parallelism: This heuristic prefers entities to play the same roles in structurally similar sentences.

(7) *"Ben went to the pub with Col on Tuesday. He played footie with him on Wednesday."*

Recency suggest that *"He"* is Col, but structural parallelism suggest that *"He"* is Ben and *"him"* is Col.

These will often give conflicting judgements. We shouldn't be too surprised. Reference resolution in general requires the use of reasoning on world knowledge (recall the example about the fascists and the council). And, even in human-to-human communication, there can remain cases that the listener cannot resolve and cases s/he resolves incorrectly.