## Semantics

Most non-transformational linguistic theories, including all the approaches being covered in STA, tackle the issue of the meaning of natural language sentences in terms of their truth conditions. The minimal 'meaning' of a declarative sentence is a truth value (note the word *minimal*; it is also recognised that a full treatment of meaning will cover much more than truth conditions, but it seems clear that it could hardly cover less.) That is, in order to know the meaning of a (declarative) sentence,[1] you need to know (at least) how the world would have to be for it to be true. In addition, these theories, and truth conditional approaches to natural language semantics in general, also assume that meaning is assigned *compositionally*, that is to say, that the meaning of a sentence, for example, is a function of the meaning of its parts.

Specifically, what this means is that expressions of, say, English are assigned set theoretic *denotations* of various sorts, which are combined constituent by constituent to give eventually the truth value associated with the sentence composed out of them.

We construct a *Model* of the world, or a state of affairs. (Hence *Model Theoretic Semantics*.) A Model is an ordered pair consisting of a set *U* -the set of entities in the world - and a function (the *Interpretation Function*) -*I* - that assigns to expressions of the language we are investigation objects constructed out of *U*.

What follows is a considerable simplification over what is required for a full treatment of natural language semantics, but it gives the flavour:

### Denotations

*Denotation* is the name given to the kind of set-theoretic object taken to be the interpretation of a linguistic expression.

a. the denotations of **names** such as *Kim* and *Sandy* are taken to be *individuals* in the Universe of Discourse which constitutes the domain of a Model of world in which the sentences are being interpreted. Since names in our treatment are **NP**s, we will take NPs to have the same denotation type, ie individuals[2]

b. **intransitive verbs** denote sets of individuals in *U* (or, more accurately, the characteristic functions of such sets[3])

c. **sentences** denote truth values, the conventional notation for which is {0, 1} (0 being 'false' and 1 'true'

Given a syntactic rule of the form,

S --> NP VP

1 *Declarative* sentences are statements. It may be that questions and commands require a different treatment.

2 This is an over-simplification. In fact, we need to assign more complex denotations to NPs in order to handle NPs like 'all programs', 'no programmers' etc.

3 Characteristic functions are those whose range is the set of truth values (conventionally represented as {0, 1}). The value of the function at an argument is 1 (true) iff the argument is a member of the set which is the function's domain. Otherwise, the value returned is 0 (false). They get their name because they *characterise* the membership of a set.

Compositionality: the meaning of a sentence is a function of the meaning of its parts.



we get the *semantic value* of S as follows:

Let $[\![S]\!]^M$, $[\![NP]\!]^M$ and $[\![VP]\!]^M$ be the *semantic values* of S, NP and VP with respect to some Model (ie state of affairs) respectively, then

$$[\![S]\!]^M = [\![VP]\!]^M ([\![NP]\!]^M)$$

That is, we get the semantic value of S in this construction by applying the semantic value of VP (which is a characteristic function) to the semantic value of NP. For example, imagine a world with two inhabitants, Kim and Sandy:

U = {Kim, Sandy}

I("Kim") = Kim
I("Sandy") = Sandy

Then $[\![$"Kim"$]\!]^M$ = I("Kim") = Kim and $[\![$"Sandy"$]\!]^M$ = I("Sandy") = Sandy.[4]

Imagine that Kim smokes and Sandy doesn't:

I("smokes") = {<Kim, 1>, <Sandy, 0>}

We define $[\![$"smokes"$]\!]^M$ = I("smokes") = {<Kim, 1>, <Sandy, 0>}(or, in other words, the set {Kim}). Thus,

$[\![$"Kim smokes"$]\!]^M$ = $[\![$"smokes"$]\!]^M$ ($[\![$"Kim"$]\!]^M$) = 1

### Logic

It is common to use logic to mediate between natural language expressions and their denotations; the technique is to translate each natural language expression into an expression of a suitable logic (here, First Order Logic) for which there exist standard techniques for assigning model theoretic interpretations. We then trade off the latter to provide an interpretation indirectly for the natural language expression.[5]

Thus
"Kim" might translate into FoL as an individual constant: k
"smokes" might translate as the function constant: *smoke'*
and
the sentence "Kim smokes" would translate as the formula: *smoke'(k)*

### Type Theory

It is also common in work on natural language semantics to use Type Theory to "glue" the different aspects of the analysis together.

4 Expressions written within double-quotes are expressions of *English*. Those written without quotes represent their semantic values.

5 Using a logic as intermediary also has the advantage that it allows us to do inferencing, using standard techniques. This is obviously useful in NLP applications.

There are two basic types: e – entities

*e* and *t* t – truth values.

In addition there is a recursive definition of a type::

if *a* is a type and *b* is a type, then <*a*, *b*> is a type

Semantically, expressions of type *e* denote individuals, and expressions of type *t* denote truth values. Expressions of type <*a*, *b*> denote functions from denotations of type *a* to denotations of type *b*.

In the logic, the type of individual constants (and variables) is *e*, the type of formulae is *t*, and the type of functions of one argument is <*e*, *t*> (ie they are functions from individual constants or variables to formulae).

In addition to the basic expressions of the logic, we have the following syntactic rule which allows us to combine the basic expressions into larger sequences

if α is an expression of type *a* and β is an expression of type <*a*, *b*>, then β(α) is an expression of type *b*

Here are some of the relevant correspondences:

| English | Category | Type | FoL | Denotation |
|---|---|---|---|---|
| "Kim" | NP | e | k | Individual eg Kim |
| "Sandy" | NP | e | s | Individual eg Sandy |
| "smokes" | V[0] | <e, t> | smoke' | characteristic function of the set of individuals that smoke |
| "smokes" | VP | <e, t> | smoke' | characteristic function of the set of individuals that smoke |
| "Kim smokes" | S | t | smoke'(k) | truth value |

To show how the syntax of English is associated with an appropriate translation we might set up the following correspondences, in which each syntactic entity of English is paired with a FoL translation:

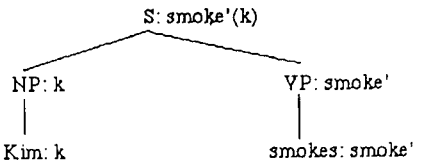
| | |
|---|---|
| S --> NP VP | Trans(S) = Trans(VP)(Trans(NP)) |
| VP --> V[0] | Trans(VP) = Trans(V[0]) |
| V[0] --> "smokes" | Trans(V[0]) = Trans("smokes")<br>Trans("smokes") = smoke' |
| NP --> "Kim" | Trans(NP) = Trans("Kim")<br>Trans("Kim") = k |



<e,t> is a function type which accepts a type 'e' as its argument, and yields type 't' as its result



Note that this way of organising the relationship between the syntax and the semantics means that, for every syntactic rule, there is a corresponding semantic rule. This approach to Natural Language semantics is known as the *Rule to Rule Hypothesis*. Putting this information together in the way specified by the rules produces what is called in Generalised Phrase Structure Grammar (GPSG) an ***interpreted tree***, in which the nodes of a standard phrase structure tree are annotated with their translations.




**Transitive verbs**

Given that the type of VP is <*e*, *t*>, with a characteristic function as its denotation, and that the type of NP is *e*, a transitive verb must be a function from NPs to VPs, that is, an expression of type <*e*, <*e*, *t*>>, whose denotation will be a function from individuals to characteristic functions of sets. The VP "likes Kim" will denote the (characteristic function of) the set of individuals that like Kim.

| English | Category | Type | FoL | Denotation |
|---|---|---|---|---|
| "likes" | V[1] | <e, <e, t>> | like' | a function from individuals to characteristic functions |

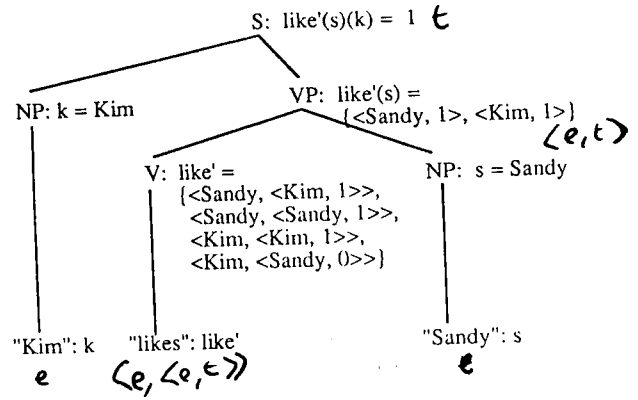
| | | | |
|---|---|---|---|
| VP | --> V[1] NP | | Trans(VP) = Trans(V[1])(Trans(NP)) |
| V[1] | --> "likes" | | Trans(V[1]) = Trans("likes")<br>Trans("likes") = like' |

In a state of affairs in which Kim likes Sandy and each individual likes herself, *like'* would have the following semantic value[6]:

$$[\![\text{like'}]\!]^M = \{\langle \text{Sandy}, \langle \text{Kim}, 1\rangle\rangle, \langle \text{Sandy}, \langle \text{Sandy}, 1\rangle\rangle, \langle \text{Kim}, \langle \text{Kim}, 1\rangle\rangle, \langle \text{Kim}, \langle \text{Sandy}, 0\rangle\rangle\}$$

The grammar would give rise to the following interpreted tree, in which I have provided *both* the logical translation *and* the semantic value for each constituent. The sentence receives the semantic value 1 (= true) with respect to this Model.

6 Observe that we are now assigning semantic values to expressions of the *logic*. English expressions inherit their interpretations *indirectly*, via the mediation of the logic.




**Lambda calculus**

*Syntax*

1. if $u$ is a variable of type $a$ and $\phi$ is an expression of type $b$, then $\lambda u[\phi]$ is an expression of type $<a, b>$.

   (This rule introduces a new kind of logical expression)

2. if $\alpha$ is an expression of type $<a, b>$ and $\beta$ is an expression of type $a$, then $\alpha(\beta)$ is an expression of type $b$.

   (this rule allows us to make use of the expressions defined in 4)

*Semantics*

Since we have now introduced variables, we need something to assign semantic values to them. This is called an *assignment of values to variables* (or, more simply, a *value assignment*) and for some opaque reason is commonly written as $g$. We therefore need to re-phrase all our statements concerning the semantic value of some expression $\alpha$, so that they are made, not only with reference to a model $M$, but also to a value assignment $g$: $[\![\alpha]\!]^{M,g}$.

The semantic counterpart of the rules above is:

4. if $u$ is a variable of type $a$, $\alpha$ is an expression of type $a$, $[\![\alpha]\!]^{M,g} = d$ and $\phi$ is an expression of type $b$, then $[\![\lambda u[\phi](\alpha)]\!]^{M,g} = [\![\phi]\!]^{M,g^{d/u}}$.

   (Where $[\![\phi]\!]^{M,g^{d/u}}$ is the result of replacing all occurrences of $u$ by $d$ in $\phi$.)

That is to say, you get the semantic value of a lambda expression applied to its argument $\alpha$, by having the value assignment $g$ assign the semantic value of $\alpha$ to all occurrences of the variable $u$ in the body of the lambda expression.

To give an example:



Let $u = x$ (type e),
let $\alpha = k$ (also type e) and
let $\phi = smoke'(x)$ (type t).

then the rules above give the following results:

$\lambda x[smoke'(x)]$ is an expression of type <e,t>

$\lambda x[smoke'(x)]$ (k) is an expression of type t ⇒ smoke'(k)

$[\![\lambda x[smoke'(x)]\ (k)]\!]^{M,g} \equiv [\![smoke'(k)]\!]^{M,g}$ (That is, the two expressions have identical truth conditions; they *entail* one-another.)

This last equivalence means that these two expression are interchangeable and licences the following:

*Lambda Abstraction*

From any expression of the form .....e.....

to an expression of the form $\lambda v[$ ....v ....] (e)

where $e$ is some expression of type $a$ and $v$ is a variable of the same type.

*Lambda Conversion*

From any expression of the form $\lambda v[$ ....v ....] (e)

to an expression of the form .....e.....

**The semantics of passives:**

Here is a PSG fragment (with features) for simple active and passive sentences:

| | |
|---|---|
| S --> NP VP | Trans(S) = Trans(VP)(Trans(NP)) |
| VP --> V[1] NP | Trans(VP) = Trans(V[1])(Trans(NP)) |
| V[1] --> "likes" | Trans(V[1]) = Trans("likes")<br>Trans("likes") = λ x[ λ y [like'(x) (y)]] |
| VP[PAS] --> V[1, PAS] PP[by] | Trans(VP) = Trans(V[1])(Trans(PP[by])) |
| V[1, PAS] --> "liked" | Trans(V[1, PAS]) = Trans("liked")<br>Trans("liked") = λ x[ λ y [like'(y) (x)]] |
| PP[by] --> P[by] NP | Trans(PP[by]) = Trans(P[by])(Trans(NP)) |
| P[by] --> "by" | Trans(P[by]) = Trans("by")<br>Trans("by") = λ x[x] (type <e, e> ) |

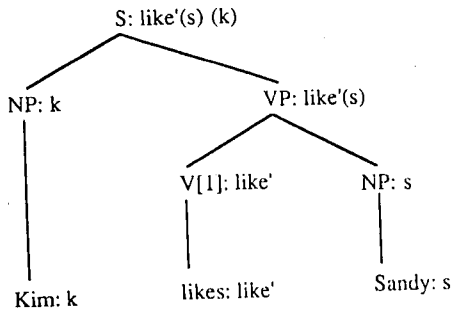
$\lambda x[\lambda y[like'(x)(y)]](k)(s)$

$\equiv \lambda y[like'(s)(y)](k)$

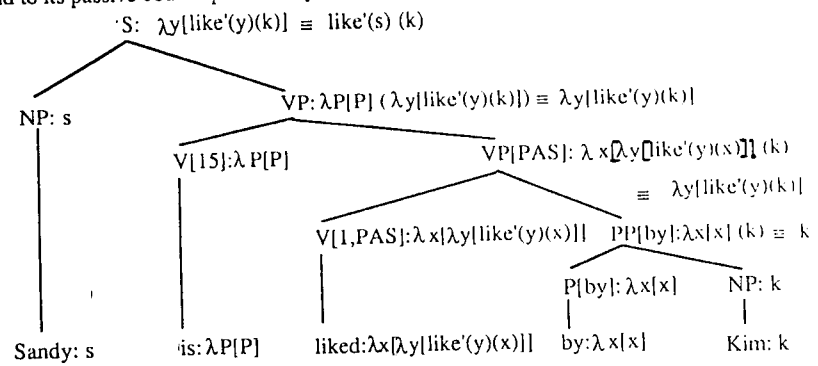
$\equiv like'(s)(k)$

| | |
|---|---|
| VP --> V[15] VP[PAS] | Trans(VP) = Trans(V[15])(Trans(VP[PAS])) |
| V[15] --> "is" | Trans(V[15]) = Trans("is")<br>Trans("is") = λ P[P] (type <<e, t>, <e, t>>) |

Here are the interpreted trees which are assigned by this grammar to the active sentence "Kim likes Sandy":




and to its passive counterpart "Sandy is liked by Kim":




Note that the root node (S) in both sentences receives the *same translation*. This, in its turn means that the two sentences will receive identical interpretations. We can thus capture the mutual entailment relations which hold between active and passive sentence pairs by means of an explicit, Model-theoretic, compositional semantics which conforms to the Rule to Rule hypothesis.

## Quantifiers

So far I have been assuming that NPs have individuals as their denotations. This, however, is an over-simplification. Consider sentences such as the following:

All men are mortal
No men are mortal

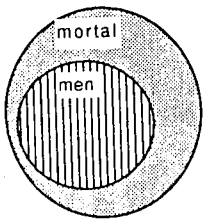




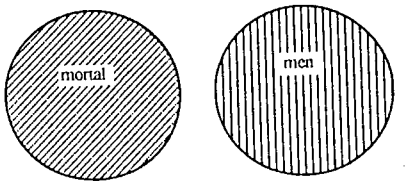
Some men are mortal

which contain the NPs "all men", "no men" and "some men". It is highly implausible to think that these NPs denote individuals. Rather, it seems more plausible to think that they involve *sets*. It standardly claimed (with considerable justification) that the truth conditions of sentences like these work in the following way.

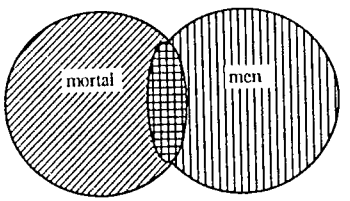
"All men are mortal" is true iff the set of men is a subset of the set of mortal things.




"No men are mortal" is true iff the intersection of the set of men with the set of mortal things is empty.




"Some men are mortal" is true iff the intersection of the set of men with the set of mortal things is *not* empty.




The standard logical translations for these sentences, which receive just these truth conditions, are the following:

| | |
|---|---|
| All men are mortal | ∀x[man'(x) ⊃ mortal'(x)] |
| No men are mortal | ~∃x[man'(x) ∧ mortal'(x)] |
| Some men are mortal | ∃x[man'(x) ∧ mortal'(x)] |

Such translations present an apparent problem for compositionality, since there is no well-formed expression in the logical translation that corresponds to the English noun phrases "all men", "no men" and "some men".

This problem has a solution, first proposed by the American logician Richard Montague (in 'The proper treatment of quantification in English'). The solution is based on the rejection of the idea that NPs denote individuals. Instead, Montague proposed to analyse their denotations as *sets of sets*. This is easily done, using Lambda Calculus.

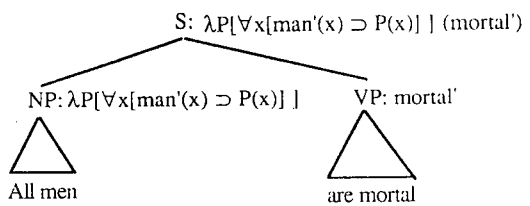
If we start with $\forall x[man'(x) \supset mortal'(x)]$, you can see that this contains two functional types (man' and mortal'). Let *P* and *Q* be variables of type <e, t>. We can then abstract over *mortal'* to give

$$\lambda P[\forall x[man'(x) \supset P(x)]\ ](mortal')$$

which, of course, has the same truth conditions. If we remove the argument expression *mortal'*, we are left with

$$\lambda P[\forall x[man'(x) \supset P(x)]\ ]$$

whose type is <<e, t> , t>: the characteristic function of a characteristic function. If you recall that a characteristic function is another way of identifying a set, what we have here, in set-theoretic terms, is an expression which denotes a set of sets – namely the set of sets of which "men" are a subset. If we consider the structure of the sentence "All men are mortal", we are driven to the following conclusion: *mortal'* is the translation of the VP "are mortal" (with "are" receiving the same translation as "is" in the passive example above - $\lambda P[P]$), which means that $\lambda P[\forall x[man'(x) \supset P(x)]\ ]$ must be the translation of "all men".




Similar reasoning gives the following translations:

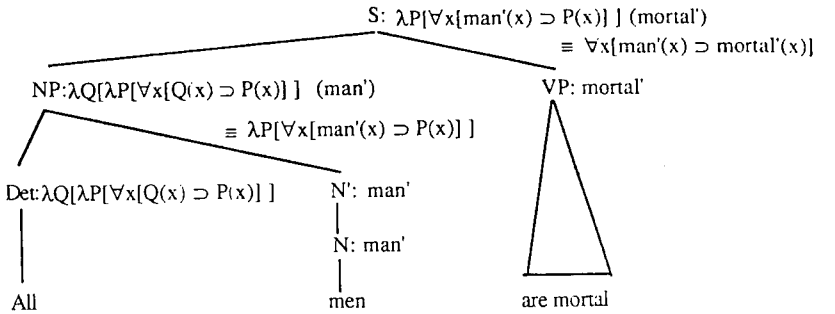
| | |
|---|---|
| No men | $\lambda P[\sim\exists x[man'(x) \wedge P(x)]]$ |
| Some men | $\lambda P[\exists x[man'(x) \wedge P(x)]]$ |

Note further that if we take these NP translations, we can continue to abstract over the remaining function constant *man'* to give:

| | |
|---|---|
| All | $\lambda Q[\lambda P[\forall x[Q(x) \supset P(x)]\ ]$ |
| No | $\lambda Q[\lambda P[\sim\exists x[Q(x) \wedge P(x)]]]$ |
| Some | $\lambda Q[\lambda P[\exists x[Q(x) \wedge P(x)]]$ |



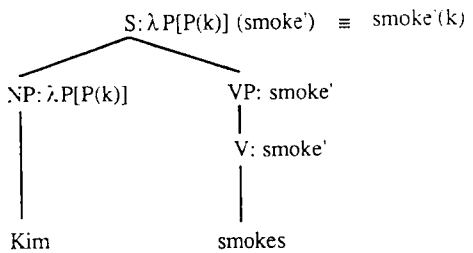
Namely, the translations of the determiners. We now have a compositional analysis of the constituents of an NP: determiners and nouns:




There remains at least one further question - what is the appropriate translation for those NPs for which a individual seemed initially to be an appropriate translation? The ingenious thing about Montague's proposal is that it extends to these NPs too, without modification. If we take the translation of a sentence such as "Kim smokes", and abstract over the function expression, instead of over the individual constant, we get the following result:

$$smoke'(k) \qquad \lambda P[P(k)]\ (smoke')$$

Removing *smoke'*, as before, leaves $\lambda P[P(k)]$ as the translation of the NP "Kim". This has the same denotation and type as "all men" etc, namely a set of sets (<<e,t>, t>). In this case, though, its semantic value is the set of sets of which Kim is a member. Lambda conversion will result in translations being assigned to sentences containing NPs of this kind as follows:




## Type Raising

There is a general point to make about the technique involved in assigning these translations to NPs. We started out in a situation in which NPs denoted individuals and VPs denoted functions from individuals to truth values. We have ended up with a situation in which NPs denote functions from VP denotations to truth values: the roles of function and argument have been reversed. The denotation of the VP has remained the same, but the NP has now become a functional type. This process (of making an argument expression into a functional expression which takes the former function as argument) is known as *Type Raising* or *Type*

*Lifting*). The general rule is as follows, if we have an expression of type $a$ and an expression of type $<a,b>$, we can 'raise' the expression of type $a$ to type $<<a, b>, b>$; an expression which takes the original functional category as argument, and returns the same value as the original function. This operation is much used in Categorial Grammar.

Note that, while this approach (which is now standardly accepted in formal approaches to linguistic semantics) deals neatly with *subject* NPS, it does not extend all that obviously to direct and indirect object NPs. If transitive verbs are of type $<e,<e, t>>$ and NPs are of type $<<e, t>, t>$, then function application of verb denotation to NP denotation is not possible. This requires some further modifications of the system, which I will not go into in any detail here. One possibility it to employ Type Raising again; this time to make direct object NPs into functions from transitive verb denotations to VP denotations (ie type <<e,<e,t>>, <e,t>>). This is adopted by some versions of Categorial Grammar, but it has the disadvantage that we no longer have a *single* type for NPs. Another approach is to change the type of transitive verbs, so that they become functions from NP denotations to VP denotations (ie type $<<e, t>, t>, <e,t>>$). This then requires some further apparatus to ensure first order reducibility.

## Scope

Formulae in FoL containing multiple quantificational expression can have different interpretations depending on the relative order of quantifiers:

| | |
|---|---|
| $\exists x[\forall y[like'(y, x)]]$ | Is true if there is some single individual in the Model who is liked by everyone |
| $\forall y[\exists x[like'(y, x)]]$ | Is true if, for each individual in the Model, there is someone or other (not necessarily the same person) that they like |

This phenomenon is known as *quantifier scope*; the left-most quantifier is said to have the other *in its scope* or to have *wide scope*. English sentences containing quantifier expressions also exhibit scope phenomena, although the situation is more complex than in logic. Many English speakers find that the readings associated with the formulae given above correspond in the following way to English sentences:

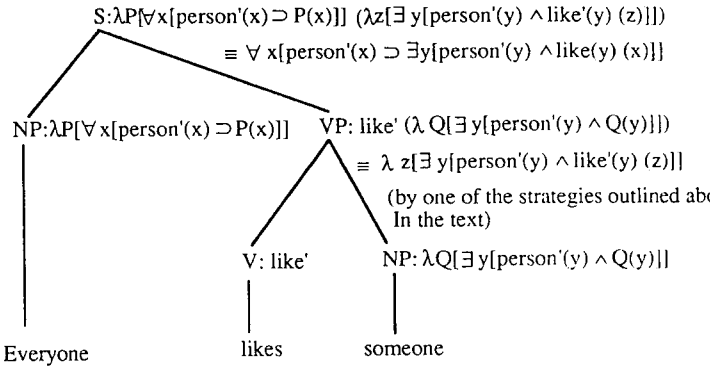
| | |
|---|---|
| $\exists x[\forall y[like'(y, x)]]$ | Someone is liked by everyone |
| $\forall y[\exists x[like'(y, x)]]$ | Everyone likes someone |

That is: the passive sentence most naturally has an existential wide scope reading (ie a unique person is liked), whereas its active counterpart more naturally has a *narrow* scope interpretation (ie some person or other). Many speakers, including me, find both sentences ambiguous, but with this pattern of readings preferred. For such speakers, the semantics of English must provide for a single sentence to be assigned *both* translations. Accomplishing this is non-trivial and there is no general agreement on how it should be done. A popular strategy is to use some form of *storage*, whereby whenever in the interpretation of a sentence an NP is encountered, there is an option to refrain from interpreting it then and there. Instead, the NP interpretation is placed in *storage* and an individual variable is substituted for it. As the process of interpretation proceeds, the option becomes available at various points to extract the NP from store and to apply it to the interpretation of the rest of the sentence. The

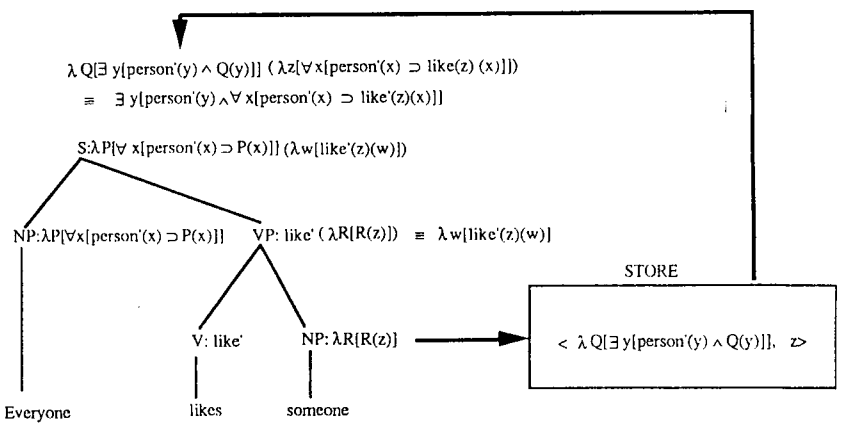


NP which is pulled out of store last has the widest scope.[7] The following two trees give the general idea.

The first one illustrates an interpretation which does not involve storage and in which the universal quantifier contained in the subject NP has wide scope:




The second illustrates the situation where the translation of the object NP, containing an existential quantifier, is put into store and is later extracted and applied to the translation of the remainder of the sentence, giving the existential quantifier wide scope.




The proper treatment of quantifier expressions presents problems of analysis and of the development of appropriate theoretical apparatus. In practical implementations, quantifier

---

7 The basic idea behind this comes from Montague who terms it 'quantifying in'. The storage implementation of the idea comes from Robin Cooper and is often termed 'Cooper storage'.

scope is a hairy problem both semantically and pragmatically, since, in a given context, not all the theoretically possible scope ambiguities[8] will actually be available.

### Intensionality

A further complication comes when we introduce temporal expressions ('Kim smoked a cigar' vs 'Kim is smoking a cigar') and modality ('Kim may be smoking a cigar' vs 'Kim must be smoking a cigar'). The evaluation of these seems to require reference not to the *actual* world or state of affairs but to some alternative one(s). In the case of temporal semantics, we need to introduce points or intervals of time into out model and to evaluate sentences with respect to them. Thus, $[\![\text{Kim is smoking a cigar}]\!]^M = 1$ iff smoke'(k, a'(cigar')) = 1 at some moment *i* identified with the moment of utterance. $[\![\text{Kim smoked a cigar}]\!]^M = 1$ with respect to some moment $j < i$ such that smoke'(k, a'(cigar')) = 1 at *j*.

Modality requires reference, not simply to alternative times, but to alternative worlds. Roughly speaking, $[\![\text{Kim may be smoking a cigar}]\!]^M = 1$ iff there is some state of affairs (possibly, but not necessarily, including the actual state of affairs) in which smoke'(k, a'(cigar')) = 1.

Some verbs of English seem crucially to involve this phenomenon, which is known as *intensionality*.[9] For example, 'There seems to be a unicorn in the garden' can be a true statement, even in a universe (like our own) in which there are no unicorns. (Whereas, 'There is a unicorn in the garden' must be false.) The relevant truth conditions seem to be that the thing being referred to as 'a unicorn' must appear to possess those properties that unicorns are supposed to have (in some alternative state of affairs).

### References


Chiercha,Gennaro andMcConnell-Ginet, Sally,1990, *Meaning and Grammar*. MIT Press (Contains a good discussion of intentionality, tense modality. Strongly recommended to anyone who wishes to pursue this issue further.)

Hobbs, Jerry R, 1986, *An algorithm for generating quantifier scopings*, Stanford University. Center for the Study of Language and Information. Report no. CSLI-86-49.

Ladusaw, W A, 1980, *Polarity sensitivity as inherent scope relations*, Indiana University Linguistics Club (Chapter 2 consists of a discussion of quantifier storage).

Partee, Barbara, Alice ter Meulen & Robert Wall, 1990, *Mathematical Models in Linguistics*, Kluwer (Part D has extensive discussion of compositionality, lambda abstraction, quantifiers, NP types and many other goodies)

Pereira, Fernando C N & Shieber, Stuart M, 1987, *Prolog and natural-language analysis*, Stanford, Center for the Study of Language and Information, Lecture notes, no. 10. (Contains Prolog examples of type raising of object NPs and of quantifier storage).


---

8 Note that any increase in the number of quantifier expressions increases the scope ambiguities commensurately: "Everyone gave something to someone". Does each person get the same thing?

9 And hence are known as intensional verbs.

# NLP Exam-Style Questions on Semantics

(i) [6 marks] Explain what is meant by each of the following terms:

a. Truth-conditional semantics.
b. Compositionality.
c. The rule-to-rule hypothesis.

(ii) [2 marks] Computational linguists have found it advantageous to use logic when giving semantics to natural languages. Why is it advantageous?

(iii) [7 marks] Assuming that you are given the chart produced by running a simple bottom-up chart parser, give detailed pseudo-code for producing *interpreted trees* from that chart. If you require there to be semantic information stored in the edges of the chart, you should make clear what the information would be.

(iv) [10 marks] Describe what is meant by *interleaved* (or *incremental*) interpretation. Discuss its advantages and disadvantages.

(v) [2 marks] Give the semantic types of the denotations of intransitive verbs, transitive verbs and ditransitive verbs. Assume that the denotations of NPs have type *e* and assume that we are not using event variables.

(vi) Explain, using NPs as an example, what is meant by type-raising. Why is it necessary in compositional model-theoretic semantics?

(vii) [3 marks] What are the consequences of type-raising the types of NPs on the types of the denotations of verbs and verb phrases.

(viii) [16 marks] You are given the following lexical entries showing word, category, semantic translation and semantic type:

| word | category | translation | type |
|---|---|---|---|
| the | Det | $\lambda Q[\lambda P[\exists x[Q(x) \wedge P(x)]]]$ | $<<e,t>,<<e,t>,t>>$ |
| loves | Vtrans | $\lambda P[\lambda y[P(\lambda x[\text{LOVE}(x,\ y)])]]$ | $<<<e,t>,t>,<e,t>>$ |

and the following grammar entries showing the rule, its semantic translation and the type of the LHS category (i.e. the "mother"):

| rule | translation | type of mother |
|---|---|---|
| NP → Det Nbar | Det′(Nbar′) | $<<e,t>,t>$ |
| VP → Vtrans NP | Vtrans′(NP′) | $<e,t>$ |

where Det′(Nbar′) denotes function application of the semantic translation of the Vtrans (Vtrans′) to the semantic translation of the NP (NP′).



Extend these two tables to include the following additional lexical entries and grammar rules:

```
Romeo:  Name      man:        N       dies : Vintrans
Juliet: Name      woman:      N       loved: Vtrans, Vpassive
a:      Det       lovestruck: Adj     was:   Aux
every:  Det                           by:    ByP
```

```
S --> NP VP                VP --> Vintrans
NP --> Name                VP --> Aux VPpassive
Nbar --> N                 VPpassive --> Vpassive ByPP
Nbar --> Adj Nbar          ByPP --> ByP NP
```

State any assumptions that you make.

(ix) [2 marks] Using your grammar and lexicon from (viii), draw the *interpreted tree* for the following sentence:

"Every lovestruck man was loved by Juliet."

(x) [8 marks] You are given the following information about types:

the type of the denotation of S is t
the type of the denotation of VP is <e, t>
the type of the denotation of NP is e

Assign semantic translation rules to each of the following syntax rules, and assign semantic translations to each of the lexical items:

```
S --> NP VP        VP --> Vintrans      VP --> Aux VPpassive
NP --> Name        VP --> Vtrans NP     VPpassive --> Vpassive ByPP
ByPP --> ByP NP
```

```
Romeo:  Name       dies : Vintrans
Juliet: Name       loves: Vtrans
by: ByP            loved: Vtrans, Vpassive
                   is:    Aux
```

State any assumptions that you make. (assume is, by, Name: e S=t)

(xi) [4 marks] Using your grammar and lexicon from (x), draw *interpreted trees* for the following sentences:

is: Aux: $\lambda P[P]$: $<<e,t>,<e,t>>$ (Maps fn. onto fn. note caps 'P' in λcalc.)

by: ByP: $\lambda x[x]$: $<e,e>$ (Maps an entity to an entity)

a. "Romeo loves Juliet."
b. "Romeo is loved by Juliet."

(xii) [2 marks] Redraw the tree you have drawn in response to question (xia) above. This time, instead of associating with each node in the tree its logic translation, associate with each node its denotation (semantic value) in the following model:

```
Universe = {ROMEO, JULIET}

Interpretation Function, I
          I("Romeo")  = ROMEO
          I("Juliet") = JULIET
          I("loves")  = { <ROMEO,  <ROMEO,  0>>,
                          <ROMEO,  <JULIET, 0>>,
                          <JULIET, <ROMEO,  1>>,
                          <JULIET, <JULIET, 1>> }
```

(xiii) [3 marks] Explain briefly what are meant by *event variables* and *case roles*.

(xiv) [3 marks] Give logical translations for the following sentence using event variables and case roles:

a. "Falstaff broke the stool with the hammer."
b. "The stool was broken by Falstaff."
c. "The stool broke."

(xv) [6 marks] Describe the two main ways of implementing selection restrictions. Compare them.

(xvi) [6 marks] Explain briefly why the second sentence in each of the following pairs of sentences would cause problems for a compositional first-order model-theoretic semantics:

(1) a. "Romeo likes Juliet."
b. "Everyone likes someone."

(2) a. "Falstaff is a green-eyed man."
b. "Falstaff is a tall man."

(3) a. "Falstaff is a green-eyed murderer."
b. "Falstaff is an alleged murderer."

(4) a. "Falstaff saw a bird."
b. "Falstaff saw a unicorn."



(5) a. "Falstaff found a unicorn."
b. "Falstaff is looking for a unicorn."

(6) a. "Falstaff likes a woman with fair hair."
b. "Falstaff wants a woman with fair hair."

The [case role] (or semantic role) of an entity is supposed to tell us the role that the entity plays in the event described by a sentence.

An [event variable] makes the event described explicit, for example Kim is the agent of a Liking event, and Sandy is the patient of the same event. More formally:

$$\exists x\,(\text{LIKE}(x) \wedge \text{AGT}(x, \text{KIM}) \wedge \text{PAT}(x, \text{SANDY}))$$

There are several distinct roles an entity can participate in (but suggested other roles, also):-

Agent - the doer of an event
Patient - the recipient of an event
Instrument - the tool, material or forces used in an event