Theories of Syntax, Semantics and Discourse Interpretation for Natural Language

Ted Briscoe
Computer Laboratory
University of Cambridge

November 11, 2013

Abstract

This handout builds on the Intro to Linguistics material by presenting formal theories of aspects of language. We mostly focus on the formalisations of aspects of the subtasks syntactic, semantic and discourse interpretation, rather than computational approximations to solve them, because it is useful to thoroughly understand the problem you are trying to solve before you start to solve it. This handout is not meant to replace textbooks – see Intro to Linguistics for readings and references herein. Please read each section in advance of the session, attempt the exercises, and be prepared to ask and answer questions on the material covered.

Contents

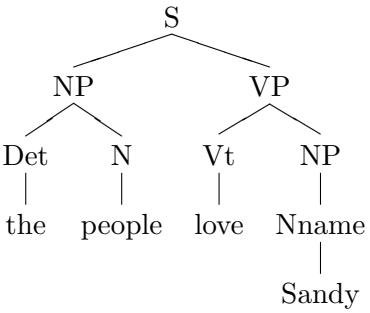
1 (Context-free) Phrase Structure Grammar

A generative grammar is a finite set of rules which define the (infinite) set of grammatical sentences of some language.

Here are some example rules for English:

a) S → NP VP
b) NP → Det N
c) NP → Nname
d) VP → Vt NP

These rules assign the sentence *The people love Sandy* the same analysis and phrase structure tree that was proposed in the Intro. to Linguistics handout, repeated below and followed by the corresponding labelled bracketing.




```
S(NP((Det The) (N people)) VP((Vt love) NP(N Sandy)))
```

Exercises

Write down the rules needed to generate this sentence from 'top to bottom'. What's missing? (easy)

The aim of a specific generative grammar is to provide a set of rules which generate (or more abstractly license, predict. etc.) all the phrase structure trees which correspond to grammatical sentences of, say, English. That is, generate all and only the word sequences which a linguist would consider correct and complete sentences considered in isolation, along with a description of their syntactic structure (phrase structure). The rules also incorporate claims about English constituent structure.

One way to formalise the grammar we have introduced above is to treat it as a context-free phrase structure grammar (CF PSG) in which each rule conforms to the following format:

Mother $\rightarrow$ Daughter$_1$ Daughter$_2$ ... Daughter$_n$

and the syntactic categories in rules are treated as atomic symbols – non-terminal symbols being clausal and phrasal categories, terminal symbols lexical categories.

CF rules encode (immediate) dominance and (immediate) precedence relations between (non-) terminal categories of the grammar. All grammars have a designated root or start symbol (see e.g. Jurafsky and Martin, ch12) for more details on CF PSGs). To make the grammar complete, we also need a lexicon in which we pair words (preterminals) with their lexical categories.

If we do formalise such rules this way, we are claiming that CF PSGs provide an appropriate (meta)theory of grammars for human languages, and thus that (all) syntactic rules for any human language can be expressed as CF PSGs.

Grammar 1 (G1) illustrates a simple CF PSG for a small fragment of English.

Grammar 1

Rules		Lexicon			
a.	S --> NP VP.	Sam	: Nname.	plays	: V.
b.	VP --> V.	Kim	: Nname.	chases	: V.
c.	VP --> V NP.	Felix	: Nname.	sings	: V.
d.	VP --> V PP.	Tweety	: Nname.	the	: Det.
e.	VP --> V NP NP.	cat	: N.	miaows	: V.
f.	NP --> Nname.	bird	: N.	a	: Det.
g.	NP --> Det N.	park	: N.	in	: P.
h.	PP --> P NP.	ball	: N.	with	: P.

Exercises

Find 10 sentences this grammar generates. Is the set of grammatical sentences generated by G1 finite? Are they all grammatical? Can you make any changes to G1 which would stop some of the ungrammatical sentences being generated? (easy)

Give a formal definition of a CF grammar (as a quintuple) and the same for a regular (right/left linear) grammar. State the additional restrictions on the right hand side of regular grammar rules over CF rules. (Hard, unless you have done formal language theory, or read some of Jurafsky and Martin (ch12) by now.)

1.1 Derivations

Given a CF PSG and lexicon, we can determine whether a sentence is or is not generated by attempting to construct a derivation (tree) for it. To construct a leftmost derivation, we start with the root symbol of the grammar and always rewrite (expand) the leftmost non-terminal category in the current sentential

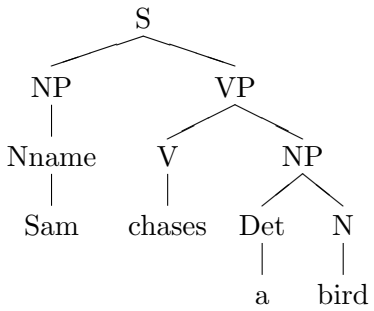
form (sequence of terminal and non-terminal categories resulting from each expansion) according to the rules in the grammar. A leftmost derivation for *Sam chases a bird* using the grammar and lexicon above is given below, where the words (preterminal categories) are given in brackets in the sentential forms:

```
S => NP VP => Nname (Sam) VP => Nname (Sam) V (chases) NP
=> Nname (Sam) V (chases) Det (a) N (bird)
```

In a rightmost derivation, we start with the root symbol but always rewrite the rightmost symbol of the sentential form. The corresponding rightmost derivation for *Sam chases a bird* is given below:

```
S => NP VP => NP V (chases) NP => NP V (chases) Det (a) N (bird)
=> Nname (Sam) V (chases) Det (a) N (bird)
```

Although the sequence of rule applications is different, the same set of rules appears in both derivations. Furthermore, the derivations are unique in that there are no alternative rewrites at any point in either which yield a derivation for the example. Constructing the derivation tree from a left/right-most derivation is straightforward – we simply represent successive rewrites by drawing lines from the mother (rewritten) symbol to the daughter (expanded) symbol(s). Both the derivations above correspond to the derivation / phrase structure tree below:



CF PSG is a so-called declarative formalism because it does not matter which order we apply rules in, we will always assign the same phrase structure tree(s) to any given sentence. Thus the rules encode the facts about grammar independently of their method of application in parsing, generation, etc.

1.2 Ambiguity

A sentence is said to be 'ambiguous' when it can be assigned two (or more) distinct semantic interpretations. A distinction is often made between two

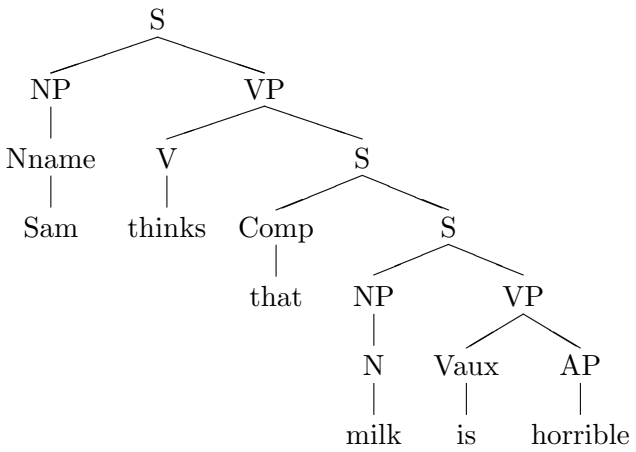
types of ambiguity, 'structural' and 'lexical'. Broadly speaking, a sentence will be said to be structurally ambiguous if a syntactic analysis of it results in the assignment of two (or more) distinct constituent structures to it, where each distinct structure corresponds to one of the possible interpretations. A sentence will be said to be lexically ambiguous, on the other hand, if it contains some ambiguous lexical item but all the distinct interpretations receive the same constituent structure. (1a) is an example of structural ambiguity and (1b) an example of lexical ambiguity.

(1) a Sam thinks that milk is horrible
    b All linguists draw trees

In (1a) the word *that* is ambiguous between a determiner reading and a complementiser reading. (A complementiser is a word that introduces a subordinate clause, traditionally called a subordinating conjunction.) The most common complementiser is *that*, but other examples include *whether* and *if*, as in (2).

(2) a Sam wonders whether Felix is in the garden
    b They want to know if Felix is in the garden

As a determiner it forms an NP constituent with the noun *milk* but as a complementiser it forms a constituent with the clause *milk is horrible*. The tree below shows the complementiser analysis where it is all milk that Sam considers to be horrible.



Another way of saying a sentence is structurally ambiguous (given a grammar) is to say that it has two or more left/right-most derivations.

Exercises

Can you construct the phrase structure tree for the other reading? With (1b) we will end up with the same phrase structure tree but different senses of the noun *trees*. Can you construct it and add the rules and words to G1 needed to generate these examples, ensuring that (1a) is assigned two analyses corresponding to the structural ambiguity? (easy)

1.3 Inadequacies of CF PSG

CF PSGs have some strengths as a theory of grammar; for instance, they seem to account for hierarchical constituency quite well and by using recursion we can capture the syntactic productivity of human language. There are infinitely many grammatical sentences of English; consider, for example, the sentence in (3).

(3) Sam saw the man in the park with the telescope on the monument.

We can introduce the recursive CF rule below

VP → VP PP

which will assign a phrase structure tree to this example which corresponds to the reading in which Sam saw the man whilst Sam was in the park using the telescope standing on the monument.

Exercises

Draw the tree by adding this rule to G1 along with some appropriate lexical entries. What rule(s) would we need to add if we wanted to capture the reading in which the telescope is mounted on a monument to be found in the park where the man is? (medium)

If you have explored G1, you will have realised that it is difficult in some cases to generate all and only the grammatical sentences. For instance, there is nothing to stop us generating the examples in (4).

(4) a *Sam chases in the park.
    b *Sam sings the cat.
    c *Sam chases the cat the bird.

We could introduce different types of verb category (eg. Vintrans, Vtrans, Vppwith, etc.) and specify each VP rule more carefully, as we did with names and common nouns, but this would lead to many more rules if we are not careful. For example, imagine what will happen if we attempt to capture person-number agreement between subject and verb in our grammar to block examples like (5)

(5) a *The cat chase the bird.
    b *The cats chases the bird.
    c *I is clever.
    d *You am clever.

If we add number and person to the categories in G1, we will end up with the much larger CF PSG, Grammar 2 (G2).

```
Grammar 2

1. S --> NPsg1 VPsg1
2. S --> NPsg2 VPsg2
3. S --> NPsg3 VPsg3
4. S --> NPpl1 VPpl1
5. S --> NPpl2 VPpl2
6. S --> NPpl3 VPpl3
7. VPsg1 --> Vsg1
...
13. VPsg1 --> Vsg1 NP
...
19. VPsg1 --> Vsg1 PP
...
25. VPsg1 --> Vsg1 NP NP
...
31. NPsg3 --> Nname
32. NPsg3 --> Detsg Nsg
33. NPpl3 --> Detpl Npl
34. PP --> P NPsg1
...
39. PP --> P NPpl3
```

We appear to be failing to directly capture a simple rule – 'the N(P) and V(P) in an S agree in num(ber) and per(son)'.

Exercises

What would happen if we also tried to subcategorise verbs into intransitive, transitive and ditransitive to avoid generating examples like **I smiled the ball to him* and combined this with the approach to agreement outlined above? How big would the new grammar be if you did this systematically? (medium)

Thinking back to the Intro. to Linguistics handout and the exercise on English auxiliary verbs, can you see how to formalise your analysis in CF PSG? How successful would this be? (hard)

1.4 Unification and Features

CFGs utilise atomic symbols which match if they are identical, unification-based phrase structure grammars (UB PSGs) utilise complex categories which match by unification. They provide us with a means to express the person-number agreement rule of English and many others more elegantly and concisely. Assume that syntactic categories are annotated with sets of feature attribute-value pairs, then we can factor out information about number and person from information about which type of category we are dealing with:

NP:[num=sg, per=3]
V:[num=pl, per=3]

where the possible values for the attribute num are sg/pl and for per 1/2/3. As well as being able to specify values for features we will also allow features to take variable values (represented as capital letters) which can be bound within a unification-based PS rule. The rules below express per-num agreement:

S → NP:[num=N, per=P] VP:[num=N, per=P]
VP:[num=N, per=P] → V:[num=N, per=P] NP:[]

Mostly, words in the lexicon will have fully specified categories but categories in rules often contain variable values, so they generalise across subcategories. Unification can be used as the operation to match categories during the construction of a phrase structure tree; that is, two categories will match if for any given attribute they do not have distinct values. The resultant category, if unification succeeds, is the one obtained by taking the union of the attributes, substituting values for variables and rebinding variables. Some examples are given below; the first two columns contain the categories to be unified and the third column contains the result of the unification.

```
a.  | NP:[per=3, num=N]   |  NP:[per=P, num=pl] |  NP:[per=3, num=pl]
b.  | NP:[per=2, num=sg]  |  NP:[per=2, num=N]  |  NP:[per=2, num=sg]
c.  | NP:[per=P, num=N]   |  NP:[per=3, num=N]  |  NP:[per=3, num=N]
d.  | NP:[per=1, num=sg]  |  NP:[per=2, num=sg] |  FAIL
    |                     |                     |
e.  |      N:[]           |         N:[]        |     N:[]
    |                     |                     |
f.  | V:[val=intrans,     |  V:[val=intrans,    |  V:[val=intrans,
    |    per=3, num=sg]   |     per=P, num=N]   |     per=3, num=sg]
g.  | VP:[in=F, out=F]    |  VP:[in=G, out=H]   |  VP:[in=I, out=I]
h.  | NP:[per=1]          |  NP:[num=pl]         |  NP:[per=1, num=pl]
```

A category consists of a category name (the functor, eg. X:) and a set of features enclosed in square brackets after the functor. Features are made up of attributes (eg. per) and values/variables (eg. 1 or P) separated by = and delimited from each other by commas.

Unification can be defined in terms of subsumption.

If two categories, A and B, unify, this yields a new category, C, defined as the smallest (most general) category subsumed by A and B, otherwise fail.

Category A subsumes category B (i.e. B is more specific than A) iff (if and only if):
1) every attribute in A is in B;
2) every attribute=value pair in A is in B;

3) every attribute=variable pair in A is either in B or B has a legal value for this attribute;
4) every attribute sharing a variable value in A, shares a (variable) value in B.

The notation I have used for categories is similar to that used for Prolog terms. However, Prolog uses fixed-arity term unification in which unifiable categories must explicitly have the same set of attributes – given this 'stronger' definition of unification case h. above would lead to FAIL because the two argument categories don't explicitly contain the attributes num or per with variable values. The advantage of 'relaxing' the definition of unification in this way is that it is notationally less verbose when categories have lots of attributes. (Jurafsky and Martin, ch15 give a more detailed introduction to unification of 'feature structures', using a slightly extended notation.)

Grammar 3 (G3) is a small UB PSG generative grammar; see if you can work out what structural descriptions it assigns to some examples and why it fails to assign any to sentences which violate per-num agreement or verb val(ence) constraints (verb subcategorisation).

```
Grammar 3

S:[] --> NP:[per=P, num=N] VP:[per=P, num=N]
VP:[per=P, num=N] --> V:[per=P, num=N, val=intrans]
VP:[per=P, num=N] --> V:[per=P, num=N, val=trans] NP:[]
VP:[per=P, num=N] --> V:[per=P, num=N, val=ditrans] NP:[] NP:[]
NP:[per=P, num=N, pronom=yes] --> N:[per=P, num=N, pronom=yes]
NP:[per=P, num=N, pronom=no] --> Det:[num=N] N:[per=P, num=N, pronom=no]

Sam N:[per=3, num=sg, pronom=yes]
I N:[per=1, num=sg, pronom=yes]
you N:[per=2, pronom=yes]
she  N:[per=3, num=sg, pronom=yes]
we  N:[per=1, num=pl, pronom=yes]
they N:[per=3, num=pl, pronom=yes]
cat N:[per=3, num=sg, pronom=no]
cats N:[per=3, num=pl, pronom=no]
sheep N:[per=3, pronom=no]
laughs V:[per=3, num=sg, val=intrans]
laugh V:[per=1, num=sg, val=intrans]
laugh V:[per=2, num=sg, val=intrans]
laugh V:[num=pl, val=intrans]
chases V:[per=3, num=sg, val=trans]
chase V:[per=1, num=sg, val=trans]
chase V:[per=2, num=sg, val=trans]
chase V:[num=pl, val=trans]
gives V:[per=3, num=sg, val=ditrans]
give V:[per=1, num=sg, val=ditrans]
give V:[per=2, num=sg, val=ditrans]
```

```
give V:[num=pl, val=ditrans]
the Det:[]
a Det:[num=sg]
those Det:[num=pl]
```

Exercises

Can you define precisely how to do a left/right-most derivation in UB PSG? Is UB PSG a declarative formalism? (i.e. does it make any difference whether we choose to do a left- or right- most derivation to the results) (easy)

Try adding the analogues of some of the other rules and/or lexical entries developed for the CF PSGs, G1 and G2 and the exercises above (e.g. names, PPs) to the UB PSG, G3. (medium)

How could we add the case=nom/acc distinction to G3 in order to block examples like **Sam chases we*? (easy)

Think again about the analysis of auxiliary verbs – how does UB PSG help make it simpler and more effective? Can you think of any remaining problems which can't be handled elegantly in the UB PSG formalism introduced? How could we make rules even simpler if we labelled head daughters or had a convention about how to select the head daughter in a rule? (hard)

2 Compositional Semantics

Pairing syntactic and semantic rules and guiding the application of semantic rules on the basis of the syntactic analysis of the sentence leads naturally to an explanation of semantic productivity, because if the syntactic rule system is recursive and finite, so will the semantic rule system be too. This organisation of grammar incorporates the principle that the meaning of a sentence (a proposition) will be a productive, rule-governed combination of the meaning of its constituents. So to get the meaning of a sentence we combine words, syntactically and semantically to form phrases, phrases to form clauses, and so on. This is known as the principle of Compositionality. If language is not (at least mostly) compositional in this way, then we cannot capture its semantic productivity.

2.1 Predicates & Arguments

In most sentences, some constituents function semantically to pick out particular individuals or (more abstract) entities and other constituents say things about these individuals or entities. For example, *Kim runs* asserts that there is an individual named Kim who runs; the sentence refers to an individual and predicates a property of that individual. In *Kim kissed Sandy* there are two referring expressions – *Kim* and *Sandy* – which pick out two individuals and

the sentence predicates a relationship between them (that of kissing at some time in the past).

Semantically predicates require different numbers of arguments. For example, predicates which ascribe properties to individuals or entities are usually one-place predicates requiring one argument. Some examples are given in (6).

(6) a Kim is asleep (asleep1 Kim1)
b Felix is boring (boring1 felix1)
c Fido smells (smell1 fido1)
d Sandy fell over (fall-over1 Sandy1)

Notice that there is no straightforward connection between types of syntactic constituents and one-place predicates. In (6) we have analysed an adjective, a verb in progressive form and a verb in present form, and a verb and preposition as one-place predicates. This decision is based solely on the semantic intuition that all these words and phrases are attributing properties to individuals or entities. Intransitive verbs and adjectives nearly always 'translate' as one-place predicates.

There are also two-place predicates of which transitive verbs, such as *kiss*, are the most common exemplars, but as with one-place predicates a variety of different types of constituent can function as two-place predicates, as (7) illustrates:

(7) a Kim likes Sandy (like1 Kim1 Sandy1)
b Sandy is in front of Kim (in-front-of1 Sandy1 Kim1)
c Sandy is fed up with Kim (fed-up-with1 Sandy1 Kim1)

Exercises

Can you think of three three-place predicates in English?
Write down their correspnoding FOL expressions. (easy)

We can treat predicates and referring expressions as predicates and individual constants of FOL. So in model-theoretic terms *Kim runs* will be true just in case the individual denoted by *Kim*, say Kim1, is in the set of individuals denoted by *run(s)*. The simple grammar below incorporates this semantics for sentences consisting of names and intransitive verbs.

a) S $\rightarrow$ NP VP : VP$'$(NP$'$).
b) NP $\rightarrow$ Name : Name$'$.
c) VP $\rightarrow$ V : (λ x (V$'$ x)).
Kim : Name : Kim1.
runs : V : run1.

(where colon separates the syntactic and semantic components of a rule and the orthography, syntax, and semantics of a lexical entry.) Since the semantics

of intransitive verbs (and VPs) will always be a function from an entity to a truth-value (or characteristic function), this will reduce to function-argument application, or a test for membership of Kim1 in the set of entities that run. We represent functions in the (implicitly) typed lambda calculus (see Jurafsky and Martin, ch15), but try to ensure that the semantics of sentences (as opposed to their subparts) are equivalent to FOL expressions. This is because, we want to use (run1 Kim1) as a FOL formula to be passed to a theorem prover (but the model-theoretic interpretation is what provides the justification for this). A sentence like *Sandy kissed Kim* requires the addition of two-place predicates for transitive verbs:

d) VP $\rightarrow$ V NP : (λ y(λ x (V′ x y)) NP′).
kissed : V : kiss1.
Sandy : Name : Sandy1.

Exercises

Write down the rule for a ditransitive verb (eg. *give*) (easy)

2.2 NP Semantics

Not all NPs are names; (8) gives some examples of NPs with determiners and common nouns. Most determiners seem to function semantically like quantifiers in FOL.

(8) a Every man smiles
b Every man likes some woman
c No man smiles

We will treat the English quantifiers *every* and *some* as analogous to the universal and existential quantifiers of FOL, respectively. The meaning of other English 'quantifiers', such as *no*, *a*, *the*, and so forth, will mostly be reducible to the meaning of these two 'basic' quantifiers. Our fragment now includes nouns, such as *man*, *woman*, and so forth. Unlike proper names, nouns do not denote individuals but rather properties of individuals. Therefore, their meaning is the same as that of an intransitive verb, such as *snore*.

If we consider the meaning of the two sentences in (9a) and (8c), it should be clear that we can't capture their meaning by translating them into the FOL expressions in b) and d) respectively.

(9) a Every man snores
b $\forall$ x snore1(x)
c Some woman smiles
d $\exists$ x smile1(x)

The problem is that the FOL expressions will be true of any individual in the model who snores or smiles. They aren't restricted in the way the English sentences are to apply to just men or just women, respectively. Obviously, we

have failed to include the meaning of the nouns in our translation. The question is how to combine the noun denotations and verb denotations correctly to arrive at the correct truth-conditions for sentences of this type. (9a) has the logical form of a conditional statement. It says that for any individual if that individual is a man then that individual snores. On the other hand, (9c) says that there exists at least one individual who is both a woman and smiles, so it has the logical form of a conjunction of propositions. Therefore, the correct logical translations of these two sentences are given in (10a,b), respectively.

(10) a $\forall$ x man1(x) $\rightarrow$ snore1(x)
b $\exists$ x woman1(x) $\wedge$ smile1(x)

To achieve this translation in a compositional fashion we will need to associate a 'template' lambda expression for the entire sentence with each of the different quantifiers – otherwise we won't be able to capture the different ways in which the NP and VP are combined semantically, depending on which quantifier is chosen. Here is the semantic translation of *every*:

every : Det : λ P (λ Q ($\forall$ x P(x) $\rightarrow$ Q(x)))

P and Q are one-place predicate variables, so the denotation of *every* is a function from one-place predicates to a function from one-place predicates to a truth-value. Here are the formulas associated with the NP *every man* and the S *every man snores*, respectively:

λ Q ($\forall$ x man1(x) $\rightarrow$ Q(x))
$\forall$ x man1(x) $\rightarrow$ snore1(x)

To maintain compositionality, we need to change the semantics of NPs consisting just of names so that they have the same type of denotation as NPs with quantifiers. Otherwise we will not be able to give a uniform semantics for the S $\rightarrow$ NP VP rule, for example, because this is supposed to combine the meaning of any NP and VP to form the meaning of an S. We can achieve this by type-raising names to functions which lambda abstract over the set of properties (one-place predicates) predicated of the name, as shown below:

NP	λ P P(Kim1)
|	|
Name	Kim1
|	
Kim	

So all NPs have the same denotation. The only difference between proper name NPs and quantified NPs will be whether the lambda abstracted set of properties is predicated of an individual constant or (bound) individual variable. The semantics of *snore* and other verbs remains the same. Combining the denotation of the NP and VP now requires applying the semantics of the NP to that of the VP, NP$'$(VP$'$), and produces the semantic analysis shown below:

Kim	snores
NP	VP
λ P P(Kim1)	λ x snore1(x)
(λ x snore1(x) Kim1)	
snore1(Kim1)	

The formulas above are logically equivalent, the third is the result of applying lambda-reduction (twice) to the first. In fact, what we have done here is combine two functions – one abstracting over predicates, the other over individuals – to produce the earlier function-argument application which we used to capture the semantics of *Kim snores*

The analysis of *Every man snores* will come out like this:

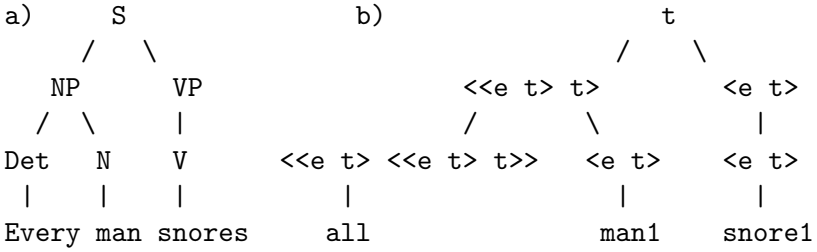
(λ P (λ Q $\forall$ x P(x) $\rightarrow$ Q(x)) (λ x man1(x)))
(λ Q $\forall$ x (λ x1 man1(x1) x) $\rightarrow$ Q(x))
λ Q $\forall$ x man1(x) $\rightarrow$ Q(x)

By two applications of lambda-reduction, we obtain the third formula. Now when we analyse *snores* we get:

(λ Q $\forall$ x man1(x) $\rightarrow$ Q(x) (λ x snore1(x)))
$\forall$ x man1(x) $\rightarrow$ snore1(x)

(see J&M p593f for Cann or more on the lambda calculus, lambda-reduction and the need for variable renaming (e.g. x to x1) to avoid variable 'collisions'.)

We can now give a more precise account of the relationship between syntactic and semantic rules and categories in the grammar. We can associate with each syntactic category of the grammar, a semantic type which specifies the type of object denoted by expressions of that category and also specifies how it combines with other types. For example, we have said that proper names denote individuals or entities (ie. any discrete object – car1, table1, Kim1 etc.). We can write this $< e >$. Sentences denote propositions (ie truth-values) written $< t >$, intransitive verbs denote functions from entities to truth-values $< e\ t >$, NPs denote functions from functions from entities to truth-values to truth-values $<< e\ t > t >$. Given this new notation we could represent the semantic analysis of *Every man snores* given above slightly more abstractly as below:



The type notation illustrates how the generic functions combine or cancel with each other to yield new functions and ultimately a truth-value. Once we know the semantic type of every syntactic category, we know how to combine them to form the semantic types of larger categories. Of course, when interpreting an actual sentence, we replace the semantic types with the functions derived from the denotations of the words in the sentence. In general, the semantics of sub-sentential constituents will be a formula of the (typed) lambda calculus, whilst the semantics of sentences will be a formula of FOL. (See Cann for more explanation. The type notation is like Categorial Grammar, which we will look at in more detail in the Advanced Syntax and Semantics module.)

Exercises

To test whether you have understood these ideas see if you can work out the semantic type of *a lecturer* in *Sam is a lecturer*, given the assumption that *be* is a purely syntactic marker here and does not contribute to the meaning of this sentence. (easy)

2.3 Scope Ambiguities

We saw that formulas of FOL containing more than one quantifier have different truth-conditional interpretations depending on the order or relative scope of these quantifiers. However, our grammar only produces one of these orderings for these and analogous sentences. For example, (11a) only receives the interpretation (10b).

(11) a Every man loves some woman

b $\forall$ x $\exists$ y man1(x) $\rightarrow$ (woman1(y) $\wedge$ love1(x y))

c $\exists$ y $\forall$ x man1(x) $\rightarrow$ (woman1(y) $\wedge$ love1(x y))

This is the interpretation which is true provided for each man there is at least one woman (who may be different in each case) who he loves. However, to get the 'every man loves Marilyn Monroe' type of reading we would need to reverse the order of the quantifiers so that the existential has (so-called) wide scope as in (11c). If you find this reading a little forced for (11a), there are other examples where it seems more natural, as (12) illustrates.

(12) a Every man loves a woman

b Every man loves one woman

c One language is spoken by everyone (here)

d A student guide showed every prospective candidate around Johns and Trinity

e There was a name tag near every door

f A flag was hanging from every window

In a) and b) it is easier to get the wide scope existential reading where there is just one woman, but most people still feel that this is not the preferred interpretation. In c), where the existential *one* occurs before *everyone*, the

existential wide scope reading seems more readily available. We adopt the narrow scope reading of the existential in d) quite readily because of our general knowledge about the organisation of the university – it's unlikely to be one student covering both colleges. In the last three examples, the existential *a* precedes the universal *every* in the surface realisation of these examples, yet the universal is given wide scope – we naturally assume there is more than one flag or name tag. We reach these conclusions on the basis of our knowledge about the relative size of flags and windows, name tags and doors, their function, and so forth.

Scope ambiguities of this type are not restricted to quantifiers. There are scope ambiguities concerning the relative scope of the negation, conjunction and disjunction operators. These also interact with quantifiers to create further ambiguities. For example, (13a) is ambiguous between b) and c), depending on the relative scope of *not* and *every*.

(13) a a) Everyone doesn't snore

b $\forall$ x $\neg$ snore1(x)

c $\neg \forall$ x snore1(x)

c) is compatible with some people snoring, whilst b) is not. There is no syntactic reason to believe that these examples are syntactically ambiguous, but nevertheless they are semantically ambiguous. This is a problem for the theory we have been developing because it predicts that sentences should only be semantically ambiguous if they are syntactically or lexically ambiguous. Thus we can produce two logical forms for *Kim sat by the bank* because we can associate *bank* with two concepts (financial institution and river side) and with *Kim hit the woman with the umbrella* because PPs can function adverbially or adjectivally (attaching to NP or VP).

The examples above force us to weaken the claim that syntax determines interpretation and undermine our initial proposal that syntactic and semantic rules are paired one-to-one in the grammar. Furthermore, nobody has come up with precise and accurate rules for resolving scope ambiguities (we saw above that this is complex and can involve word order, the meaning of the rest of the sentence, prosody or intonation, and general world knowledge. (For more on (generalised quantifiers, compositionality, etc see J&M p626f or Cann's book – we will look at recent statistically-based approaches to resolving scope in Advanced Syntax and Semantics.)

Exercises

Does anything analogous occur in the semantics of logics or programming languages?

2.4 Intensionality

Embedded propositions cause problems for our semantics. For instance, *believe* takes a sentential complement, as in *Kim believes (that) Sandy loves Fred* which we might 'translate' into:

believe1(Kim1, love1(Sandy1 Fred1))

However, there is a problem with this analysis because it is quite possible for the two examples in (14) to correctly characterise Kim's beliefs.

(14) a Kim believes Sandy loves Fred

b Kim doesn't believe Miss UK loves Fred

If Sandy is Miss UK, then we must assume that Kim is unaware of this, otherwise he would modify his beliefs appropriately. The translation of *Miss UK* will be something like ∃ x miss-uk1(x) ..., so in a world where Sandy is Miss UK, the referent of this NP and *Sandy* will be Sandy1 and we will end up saying that Kim both does and doesn't believe that Sandy1 loves Fred. The way out of this problem is to say that Kim believes the proposition *Miss UK loves Fred* and not its denotation (ie. its truth-value t/f) in the actual world. Or in other words, that Kim's belief is about the sense of the embedded sentence. This example shows then that not all meaning can be reduced to the extended notion of reference that we have called denotation.

One way of approaching this problem is to switch from an extensional logic whose model-theoretic interpretation concerns only actual states of affairs to an intensional logic whose model-theoretic semantics is characterised in terms of 'possible worlds'. In this approach, the extension of an expression can vary depending on which world we are considering, so in the actual world the extension of *Miss UK* may be Sandy1 but in another possible world it may be different. We will define the intension of an expression as the set of extensions that it has in every possible world. Names are usually treated as having the same extension in every possible world. Since the extensions of *Sandy* and *Miss UK* will differ between possible worlds, the intensions or propositions conveyed by these two embedded sentences will also differ. The intension of a proposition will be the set of truth-value assignments defined by evaluating its truth in every possible world. Thus, if Kim's beliefs are about the intensions of the embedded sentences, they will be logical (rather than contradictory) but possibly inaccurate in the actual world. All verbs which take embedded sentences as arguments denote some relation between an individual and (the intension of) a proposition.

This analysis of embedded propositions predicts that examples such as *Kim believes Miss UK is not a feminist* should be ambiguous. It could convey a proposition telling us that Kim believes that Sandy (who she knows is Miss UK) is not a feminist or one telling us that Kim believes that whoever Miss UK is she is not a feminist. These two alternative logical forms can be notated as:

a) ∃ x Miss-uk1(x) ∧ Believe1(Kim1 ^[¬ Feminist(x)]
b) Believe1(Kim1 ^[∃ x Miss-uk1(x) ∧ ¬ Feminist(x)])

The 'hat' sign indicates that what is believed is the intension (rather than extension) of the formula in square brackets. If the existential variable is in the scope of the intension operator, we get the so-called *de dicto* reading where Kim believes that whoever Miss UK is, she is not a feminist. If the existential

has wide scope, then we get the so-called *de re* reading where Kim believes a particular proposition about Sandy (or whoever happens to be the extension of *Miss UK* in the actual world).

However, one consequence of this analysis of *believe* is that if Kim believes any necessarily true propositions, then Kim must believe every necessarily true proposition. For example, if (15a) is true, then b) must be true (if we assume that *bachelor* just means 'unmarried man').

(15) a Kim believes it is raining or it isn't raining

b Kim believes all bachelors are unmarried men

However, this seems wrong because it is not a consequence of Kim's beliefs about rain that he believes things about bachelors. To see why we get into this bind, it is necessary to think about the denotation of a proposition in possible world semantics. The intension of a proposition is just a set of truth-value assignments derived by evaluating its truth in every possible world. However, every necessarily true proposition will be true in all worlds, so their intensions will be identical.

It seems that the notion of intension defined by possible world semantics is not adequate to capture the sense (as opposed to reference) of a linguistic expression. In any case, a direct interpretation of possible world semantics in a computational context would be difficult. (See Cann's book for more on possible worlds, intensionality, etc.)

Exercises

Can you see why? – How many possible worlds are there? How many facts are there in one world?

Instead we will ignore these problems and do theorem proving with the FOL formulas for these examples. This works (to some extent) because we are keeping propositions distinct by utilising their syntactic form – believe1 (Kim1 love1(Kim1 Sandy1)) is structurally distinct from $\exists$ x miss-uk1(x) $\wedge$ believe1(Kim1 (love1 Kim1 x)). Why is this not really adequate?

2.5 Word Meaning

So far we have said very little about word meaning. However, we have made a couple of assumptions in order to get our account of sentence meaning off the ground. The first assumption is that word meaning divides into a number of distinct senses (much as in a conventional dictionary). So we have written Snore1 as the translation of the first (main) sense of *snore*. In cases of lexical ambiguity, we can create further senses – Bank1, Bank2, etc. This approach is fine providing that we agree that word senses are really distinct in this way. However, many words exhibit polysemy; that is, they have a variety of closely related meanings which shade off into each other. Consider, for example, the meaning of *strike* in (16).

(16) a Kim struck Sandy
b Kim struck a match
c Kim struck a bargain
d Kim struck Sandy as ridiculous

Different dictionaries will make different decisions on how to 'divide up' the meaning of *strike* into distinct senses, underlining the rather arbitrary nature of this activity. Truth-conditional semantics offers no insights into this issue, so we will not dwell on it further. However, it is important to remember that when we discuss word meaning, we are discussing word sense meaning (or equivalently, the meaning of a predicate or lexical item).

The second assumption is central to the principle of compositionality and the account of semantic productivity which is incorporated into truth-conditional semantics. The contribution of word meaning to sentence meaning must take the form of an invariant contribution to the truth-conditions of each sentence. Otherwise, compositionality will be undermined, because the rules which combine the meanings of words, phrases and clauses to form sentence meanings do not modify the units which they combine. Of course, this does not prevent selection of an appropriate sense in a particular context, but it does imply that word meaning is not unpredictably altered by context. Some linguists argue that word meaning is partially determined by the context of occurrence – idioms and metaphors usually figure largely in this type of debate. For example, in (17) it is implausible to argue that productive semantic rules combining the meanings of the individual words produce the most likely interpretation.

(17) a Truth-conditional semantics is dead wood
b Kim is rapidly sliding into a moral cesspool

How serious a problem you think this is will depend a) on how common and central to language you feel idiomatic and metaphorical usage to be and b) on whether you believe a theory of idioms and metaphors can be derived from a theory of 'literal' meaning.

2.6 Theorem Proving

So far, we have not considered the computational implementation of a truth-conditional approach to semantics. We cannot transport model theory (or proof theory) onto a machine directly, because of the problems connected with completeness of the model and with obtaining just those inferences we want (as opposed to an infinite number of mostly useless ones). For example, in model-theoretic semantics we can characterise the truth of *Kim doesn't love Sandy* relative to some model on the basis of the denotation of love1 not containing the ordered pair <Kim1 Sandy1>. However, this only works if the model is *complete* in the sense that it represents every fact (and non-fact) about the world. It is not practical to implement complete, closed models. For this reason, automated theorem proving takes place in the context of a database of known facts represented as formulas in some logic (ie. a partial model). Fur-

thermore, it is not possible to apply proof-theoretic rules in an unconstrained way to such a database.

Exercises

What would happen if you freely applied the rule of $\wedge$-Introduction (p, q $\models$ p $\wedge$ q). (easy)

The techniques of automated theorem proving provide ways of applying a subset of proof-theoretic rules in a constrained and goal-directed way. Mostly, such techniques are restricted to a subset of FOL, so it follows that a computational implementation of NL semantics will need to (at least) restrict itself to FOL analyses.

Our approach will look something like this: NL Semantics $\rightarrow$ [Lambda-reduction] $\rightarrow$ FOL $\rightarrow$ [Skolemisation, etc.] $\rightarrow$ Clausal Form $\rightarrow$ Query/Assert in Database.

Forward chaining is a technique for doing goal-directed inference. It is a technique for implementing Modus (Ponendo) Ponens (MPP) or implication elimination (p $\rightarrow$ q, p $\models$ q) which doesn't result in making many 'unnecessary' inferences. For example, if we store (18a) in our database and apply MPP and Universal Elimination, we will infer as many formulas like (18b) as there are men in the database and instantiate (18a) to as many useless formulas like c) as there are individuals (who are not men) in the database.

(18) a (all (x) (if (man1 x) (snore1 x)))
b (snore1 Kim1)
c (if (man1 felix1) (snore1 felix1))
d (man1 Kim1)

Forward chaining is a technique which dispenses with the step of Universal Elimination and inferring formulas like c). Instead, formulas like a) have no effect until an assertion like d) is added to the database and only then is b) inferred. To do this formulas like a) must be represented in implicit-quantifier form:

(if (man1 _X) (snore1 _X))

Now we replace UE with unification (i.e. term matching, see section 1.4 and J&M, ch15 and ch18) and state the forward chaining rule as follows: from p$'$ and p $\rightarrow$ q infer q$'$ where p$'$ unifies with p and q$'$ is the result of making the same substitution(s) in q. Thus we have converted UE + MPP into unification and search in a database of formulas expressed in a notational variant of FOL.

Most of the time forward chaining is still too undirected to be efficient. The effect of adding any universally quantified statement to the database will be to also add all the valid inferences which follow from it (by UE + MPP) in the database. Backward chaining is an alternative technique which performs these inferences at 'query time' rather than 'assertion time'. That is, inferences are only performed when they are needed.

For example, if we query the database with (19a) and the database contains b)

and c), then backward chaining will attempt to unify a) with all the variable-free, ground propositions in the database and, if this fails, with all the consequents of implications like b). The latter will succeed, but before we can answer 'yes', it is necessary to prove the truth of the antecedent with identical variable substitutions. This can be done by unifying the resulting variable-free proposition with the identical formula c).

(19) a (snore1 Kim1)
b (if (man1 _X) (snore1 _X)
c (man1 Kim1)

We need to distinguish queries from assertions because the variables in the implicitly-quantified queries behave more like existential than universal variables. So the alternative query (20) means 'is there a value for _X which results in a true proposition?'

(20) (snore1 _Y)

In this case backward chaining will produce the sub-query (man1 _Y) which will unify with the ground proposition, again signifying success with _Y=Kim1. Intuitively, querying with a variable-free formula is like a yes/no-question, whilst one containing variables is like a wh-question.

We can describe backward chaining more precisely now as follows: Given Query: q' where $p \rightarrow q$ is in the database and q' and q unify with substitutions t, then Query: p^t (ie. the result of making the same substitutions in p) and if this unifies with p'' with substitutions t' then $t \sqcap t'$ is the answer.

There is much more to theorem proving than this; eg. negation as failure, skolemisation, etc (see J&M and references therein).

3 Discourse Processing

The problem with the truth-conditional semantics plus theorem proving approach to language understanding is that we have no notion of the goals of the speaker/user, what speech acts s/he is attempting; for example, asserting, requesting, asking. The system will simply assume that all declarative sentences are assertions and all interrogative sentences are requests for information.

Exercises

Can you construct a situation / dialogue in which this would lead to a system misinterpreting some input?

Below I survey some ways of augmenting deduction (entailment) with techniques able to handle some of the phenomena described in the Intro to Linguistics handout.

3.1 Abductive Inference

A form of plausible inference which often looks like the 'structural opposite of MPP' and involves reasoning from consequent to antecedent, as in (21).

(21) a (if (drunk x) (not (walk-straight x)))
b (not (walk-straight john))
c (drunk john)

This is not deduction and the truth of the 'conclusion' c) is not guaranteed by the truth of the premises a) and b). It is often an inference about causation to find explanations for events. Various forms of abductive inference have been proposed as a way of recognising speaker intentions or goals and modelling the defeasible nature of these inferences. For example, the discourse in (22a) can be interpreted using abductive inferences.

(22) a Kim can open the safe. He knows the combination.
b (can Kim1 (open safe1))
c (know he (combination-of x c))
d (if (can x state) (know x (cause (act x) state)))
e (if (combination-of x c) (cause (dial z x c) (open x)))
f (plausibly-if (and (know x p) (if p q)) (know x q))
g (know Kim1 (cause (act Kim1) (open safe1)))
h (know he (cause (dial z x c) (open x)))
i (know Kim1 (cause (dial Kim1 safe1 c) (open safe1)))

I have given a FOL-like representation (in which variables are implicitly universally quantified) of how to do this. b) and c) are the formulas associated with the two sentences (by parsing / semantic interpretation). d), e) and f) are background knowledge. g) is inferred from b) and d) by MPP / forward chaining. h) is inferred from c), e) and f), where e) is first 'embedded in' f) by replacing the propositional variables p and q by e). i) is derived by matching g) and h) and represents recognition of the elaboration relation, because 'dial' is more specific than 'act' and Kim1 more specific than 'he'. The details of the matching process are complex, and must involve more than first-order unification of (individual) variables. A 'side-effect' is that the antecedent of the pronoun is found, as is the link between *the combination* and safe1.

Exercises

Can you see any disadvantages to this technique as a general approach to discourse understanding?

3.2 Scripts and Plans

Scripts can supply default inferences, aid pronoun resolution, etc. In fact they are just another way of representing the world knowledge necessary to do the sorts of reasoning required in discourse processing. For example, here is a 'restaurant' script:

```
restaurant : X enters R
             X sits at table
             X reads menu
             X orders food from W
             W brings food
             X eats food
             W brings invoice
             X pays W
             X leaves R
```

This looks fairly trivial, but can be used to resolve sense ambiguities, eg. *bill* (= invoice), pronoun reference *he eats quickly* (= X eats), infer that certain things have happened even though they haven't been mentioned eg. *he examined the menu* (X is sitting at a table), deal with definite reference like *the bill*, *the menu*, etc. Obviously, a useful script representation would need to use a richer notation than that above.

Problems with scripts arise with temporal and sequential reasoning (matching bits of dialogue to relevant bits of script), script selection (a business lunch), and script decomposition (arbitrary depending on focus of attention). One way to get round these problems with scripts (hopefully) is to hang on to the notion of a stereotypical schema but give up the idea of analysis by matching (into a script) and instead try to dynamically construct the appropriate script from 'smaller' units representing individual actions, states, and goals.

3.3 Shallow, Knowledge-Poor Anaphora Resolution

The approaches outlined in the previous sections all require a system to be capable of sophisticated inference over, in principle, large quantities of wide ranging information. Therefore, systems of this kind are restricted to very circumscribed domains and, even so, produce extremely brittle performance. Such systems are quite inappropriate, for example, for tasks like information extraction from documents, say, on the web, where neither language nor domain is circumscribed. However, a task like information extraction crucially requires the tracking of anaphoric links to accrue information about discourse referents. For example, if a user is interested in finding news reports discussing President Clinton's impeachment defense, a system returning all documents containing these words will require the user to sort through many irrelevant documents (low precision). For example, documents containing sentences like *President*

Clinton's decision to use cruise missiles in the Gulf was undoubtedly motivated by his twin desires to avoid vote-losing loss of American life and to distract the American public from the weakness of his defence against impeachment for perjury. are unlikely to be very relevant. A relevant document, on the other hand, might contain sentences like *The president will today appear before the Senate judicial committee to answer their questions concerning the Monica Lewinsky case. His defense is likely to be that he did not knowingly commit perjury in answering earlier questions about the affair, though his answers did not reveal full details of the relationship.* The relevance of this document might be determined if *president*, *His*, *he* and *the affair* could all be linked to the discourse referent 'Clinton'.

One way to try to robustly recognise discourse referents and their coreference links in a documents is to firstly identify all the NPs in a document, secondly to identify their grammatical relation to the verb in their clause (subject, object, indirect object, temporal adjunct, etc), thirdly to record their relative positions in the document, and fourthly, to identify the discourse segment in which they occur. The first and second might be achieved by syntactic parsing, the third by simply indexing the linear position of each identified NP, and the fourth approximated by making use of superficial textual clues such as paragraph boundaries and cue words for discourse segment boundaries (e.g. *anyway*).

The coreference resolution procedure forms a set of coreference classes by stepping through the NPs 'from-left-to-right', according to their positional index, and either assigning them to a new or existing class. Initially, impossible links are pruned on syntactic and morphological grounds; for example, if the current NP does not agree in number or gender with a coreference class or occurs inside a NP coreferential with a class. Then the link to one of the remaining existing classes is determined by a salience weight which integrates various factors as a weighted sum, representing the relative salience of the class on the basis of the salience of the existing member NPs. This is recalculated whenever the current NP is determined to be coreferential with a class. The salience factors and their weights are listed below:

1) Discourse Segment (DS) Recency (50): iff current NP is in current DS
2) Sentence Recency (100): iff current NP is in current sentence
3) Subject Emphasis (80): iff current NP is subject
4) Existential Emphasis (70): iff current NP is pivot of existential sent.
5) Possessive Emphasis (65): iff current NP is possessive
6) Accusative Emphasis (50): iff current NP is direct object
7) Dative Emphasis (40): iff current NP is indirect object
8) Oblique Emphasis (30): iff current NP is contained in a NP
9) Head Emphasis (80): iff current NP is not contained in a NP
10) Argument Emphasis (50): iff current NP not in an adjunct

The class with greatest salience is always chosen as coreferential. Adding a new NP to a class always increases its salience weight and thus the likelihood of further coreference to this class.

Testing algorithms like this on documents from various genres and domains extracted from the web has produced accuracy rates between 75%-85% correctly resolved NPs over all NPs in a document. (For further discussion of anaphora resolution see Jurafsky and Martin, ch21.)

Once a set of coreference classes has been discovered, a document can be mapped to a structure which represents what has been predicated of any given discourse referent, This provides a more satisfactory starting point for finding documents relevant to user queries, or for summarisation. Much of current NLP research is focussed on sidestepping the 'knowledge bottleneck' which results from the knowledge-based perspective on NLP by using 'shallower' statistical approximations determined from empirical work with large corpora.