Natural Language Processing

8 Lectures, 2001

ejb@cl.cam.ac.uk
http://www.cl.cam.ac.uk/users/ejb/

1 Lecture Synopsis

Aims

To introduce the fundamental techniques of natural language processing, develop an understanding of the limits of those techniques and of current research issues, and evaluate some current and potential applications.

- **Introduction.** Brief history of NLP research, current applications, the state-of-the-art, knowledge-based *versus* statistical approaches. (section 3)
- **Words.** Parts-of-speech and their finite-state disambiguation, inflectional and derivational morphology, finite-state spelling rules, ambiguity and semi-productivity. (section 4)
- **Syntax.** Generative grammar, constituency, ambiguity, context-free grammars, descriptive adequacy, a simple unification-based grammar, where are NLs on the Chomsky hierarchy? (section 5)
- **Parsing.** (Non-)deterministic parsing, parsing complexity, parsing preferences (garden paths), shift-reduce and chart parsing. (section 6)
- **Semantics.** Truth-conditional semantics, compositionality, syntactically-driven semantics, scope ambiguities, intensionality. (section 7)
- **Statistical Parsing.** Coping with ambiguity and change/variation, probabilistic grammar, grammar induction, lexical approaches and sparse data. (section 8)
- **Understanding (discourse).** Theorem proving, speech acts, reference, resolving anaphora, discourse structure, abductive inference, and planning. (section 9)
- **Applications of NLP.** Machine translation, information retrieval / extraction, spoken language understanding, text-to-speech synthesis. (section 3.2)

Objectives

At the end of the course students should:

- Be able to describe the architecture of and basic design for a generic NLP system 'shell' for a central task, such as mapping text to appropriate logical representations.
- Be able to discuss the current and likely future performance of several NLP applications, such as machine translation or information retrieval.
- Be able to briefly describe a fundamental technique for processing language for each of the main subtasks, such as morphological analysis, syntactic parsing, etc (as indicated in the lecture synopses).
- Understand how these techniques draw on and relate to other areas of (theoretical) computer science, such as formal language theory, formal semantics of programming languages, or theorem proving.
- Be able to compare and evaluate several approaches to some subtasks, such as statistical vs. knowledge-based part-of-speech disambiguation or anaphora resolution.
- Recognise the major properties of natural languages and how these relate to and differ from those of artificial languages.

I'll start by summarising the history of NLP, discuss the state-of-the-art, and then different ways of thinking about human natural language (as opposed to an artificial invented language, such as a logic or programming language). During the middle six lectures I will describe the various problems which need to be solved in order to develop NLP systems and some current approaches to them. I'll try to point out the weaknesses of the dominant knowledge-based approach to NLP and show how statistical approaches might resolve some of them. In the last lecture, I'll describe and evaluate some NLP application systems and discuss to what extent the same types of knowledge about language are important to apparently quite different tasks.

1.1 Recommended Reading

Background:

Pinker, S. *The Language Instinct*, Penguin, 1994.

An entertaining introduction to linguistics

Yule, G. *The Study of Language.* Cambridge University Press, 1985.

One of many simple introductions to linguistic concepts like *inflectional morphology...*

Burton-Roberts, N. *Analysing Sentences*, Longman, 1986.

One of many good introductory books on grammatical analysis.

Rayward-Smith, V. *A First Course in Formal Language Theory*, Blackwell, 1983.

A simple introduction to formal language theory, automata theory and the Chomsky hierarchy.

Recommended Books:

All of these cover NLP/Computational Linguistics. 2) is the best overall. 1) provides good recent overview, 2) covers the material in lectures 5-7 best, whilst 3) is better for 2-4. 2, 6 and especially 7 discuss statistical NLP in detail. 8) is a new and generally good comprehensive textbook which also covers speech processing. Any one of them will provide you with an expanded account of most of the topics I discuss. I also recommend specific readings in the text.

1) Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* Prentice-Hall. (Especially Chapter VII, but see III, IV and V for supporting material.)

2) Allen, J. *Natural Language Understanding*, Benjamins, 1987/1993. (2nd ed. much better)

3) Gazdar, G. & Mellish, C. *Natural Language Processing in Lisp/Prolog*, Addison-Wesley, 1989.

4) Pereira, F. & Shieber, S *Prolog and Natural-Language Analysis* CSLI Series, Univ of Chicago Press, 1987.

5) Winograd, T. *Language as a Cognitive Process, vol. 1, Syntax*, Addison-Wesley, 1983.

6) Charniak, E. *Statistical Language Learning* MIT Press, 1993.

7) Manning, C. and Schutze, H. *Foundations of Statistical Natural Language Processing* MIT Press, 1999.

8) Jurafsky, D. and Martin, J. *Speech and Language Processing* Prentice-Hall, 2000.

Acronyms / Terminology etc.

I make an effort to define everything in the text on first use etc. However, you may find the following helpful (in the dept. library):

Trask, R.L. *A Dictionary of Grammatical Terms in Linguistics*, Routledge, 1993.

Errors / Putrid Prose etc.

I very much appreciate feedback on how to improve this handout, by email if possible...

2 Study and Supervision Guide

The course attempts to survey a broad field in a short time. Although there are no formal prerequisites, the strategy is to outline how tools and techniques familiar from mainstream computer science can be deployed for NLP. The following CST courses are particularly relevant: *Regular Languages and Finite Automata*, *Probability*, *Logic and Proof* and *Artificial Intelligence*. Most of the background needed can be found in Russell & Norvig's recent introduction to artificial intelligence, which is on the recommended reading list. With this, it should be possible to see how the techniques outlined in the handout can be formalised, implemented and extended. As it is necessary to know something about language to understand NLP and as most computer scientists won't have come across linguistics, the handout goes into somewhat more detail on basic linguistic terms, grammatical analysis, and so forth. Nevertheless, both potential supervisors and students would benefit from reading an introduction to linguistics. The recommended background reading by Pinker is entertaining and good, but something more detailed on grammatical analysis like the book by Burton-Roberts will be useful, especially for anyone who did not study grammar at school.

The handout is not intended to replace textbooks. On the contrary, many of the questions it contains are likely to require further reading before they can be answered satisfactorily, though I will also answer most of them in lectures. I make assumptions about background knowledge which will not be completely accurate in any individual case. Probably, the best approach to using the handout is to read the relevant section quickly prior to the lecture and bring it along so that you can quickly determine which subsection I am elucidating or expanding on. If after I have done so, some particular part of the handout or lecture is still unclear, it is a good idea to ask me in the lecture to try again, or mark that section for discussion in a tutorial, or as something to read about in a textbook.

The handout contains many questions which are intended to test understanding of the material, provoke further critical thought etc. A reasonable (though fairly reactive) way of running and/or preparing for supervisions would be to go through these questions. Quite a few of these questions can be given better answers on a second / revision reading through the handout after the lecture course is over. The exam questions for the last three years give a good indication of the types of question which might come up, though the course is continuously and incrementally updated. (Solution notes are available from A201.) The handout is also interspersed with more specific non-introductory references which are mostly more appropriate for supervisors to use to learn more about particular techniques, but can be used by students who have read further material in one of the introductory texts and still want to pursue a topic further.

3 Introduction

3.1 Potted History of NLP

NLP research began in the USA in the early fifties when it became a priority to develop machine translation (MT) systems. (These were mostly Russian-English and were motivated by the desire to translate large numbers of Russian scientific documents. The US felt it was losing the aviation / space race at the time.) The systems developed were mostly little more than automated bilingual dictionaries which attempted translation through word-by-word substitution. Can you think of, say, an English sentence which would not translate successfully into French (or vice versa) this way? Although results seemed impressive initially they didn't get any better and much of the funding was withdrawn – they worked out that, given the expense of computing at the time, it was cheaper to pay people to do the translations.

NLP research continued in US universities on a smaller scale through the sixties and began to be influenced by ideas emanating from linguistics. Noam Chomsky had 'revolutionised' linguistics with his theory of generative transformational grammar which offered the promise of a formal, mathematical approach to grammar and made linguistics very fashionable (for a while). NLP researchers saw the incorporation of grammar into MT as a solution to some of the problems with the word-by-word approach. Consequently, much of the focus of research shifted to parsing – the deployment of grammatical information to automatically construct a grammatical (or syntactic) analysis of a sentence. However, it wasn't until the early seventies with the appearance of Terry Winograd's SHRDLU program that the promise of NLP was generally appreciated.

SHRDLU is a system capable of participating in a dialogue about a microworld (the blocks world) and manipulating this world according to commands issued in English by the user. (SHRDLU is widely documented, for instance in Winston, P. *Artificial Intelligence*, Winograd, T. *Natural Language Understanding*.) This system appeared to demonstrate that genuine language understanding by computer was attainable, but much of the intervening period has been spent attempting to put Winograd's achievements on a more theoretically sound platform and to generalise them to more complex domains and language. It's arguable that today, whilst we can build more elegant and wider-coverage systems more quickly, no qualitative jump has been made in text or dialogue understanding.

During the eighties and nineties, NLP work has been fairly well-funded in America, Europe and Japan and the outlook remains broadly optimistic. There have been significant advances in the areas of parsing and semantic interpretation – working out how word meaning should be combined to form the literal meaning of a sentence. However, many difficult problems remain. Probably one of the main reasons why the field continues to be funded is because the economic factors have changed completely since the fifties. For example, the EU spends millions of pounds on human translation each year, whilst the cost of computing has decreased dramatically. At the same time, the wider availability of computers and the decreasing technical sophistication of users make factors such as the user interface, ease of data acquisition, etc more important, and NLP offers potential solutions to such problems.

3.2 NLP Applications

Roughly, NLP applications include tasks which utilise (or should utilise) linguistic knowledge; these range from extensions to word processing such as hyphenation, spelling and grammar checking / correction through to natural language interfaces to database systems, information retrieval and extraction, MT, speech recognition/understanding systems, and so forth. Commercial products for all the applications listed below exist (and more), however, its certainly true that none work 100% successfully and probably true of all the more ambitious tasks that their limited performance restricts their utility to the extent that only a tiny fraction of the potential market for such products has been reached. Below I briefly summarise the state-of-the-art performance for the main tasks:

NL Interfaces to Software Systems: These include interfaces to databases, help documentation, expert systems, operating systems, integrated office software, etc. Most commercial products are interfaces to databases / DBMS packages. They were sold in the 1970s as expensive independent packages which required customising to a particular system and database application. There was a brief trend in the 1980s to sell DBMS packages with simple, customisable NL interfaces where the user creates his own appropriate specialised vocabulary, and so forth, and use the interface as an additional selling point, but these didn't catch on as users generally preferred to learn the formal query language. Can you think of any reasons why this might be the case?

The best (prototype) systems have vocabularies of a few thousand words and quite impressive but nevertheless limited grammatical coverage. They will correctly interpret quite complex queries involving quantifers, modifiers, coordination, etc. – eg. *Show each of the annual sales figures for 1989 for every product in the XR and YR ranges* – but will often fail on extended dialogues in which the user is tempted to rely on previous context to keep his requests short – eg. *Now show me them for the ZRs.*
You can have a dialogue with CHAT/TIPS at:
`telnet://debra.dgbt.doc.ca:3000`

Machine Translation: There are quite a few commercial systems all of which require some manual pre- and/or post-editing or interaction with the user. The biggest have vocabularies of tens of thousands of words and are capable of translating between several different language pairs. They make use of lexical and syntactic information in both languages and incorporate distinct 'transfer' components for each language pair covered (that is, they don't attempt to translate each language into an 'interlingua' – a language-independent unambiguous representation of meaning – but rather define mappings between pairs of more superficial and language-specific representations). Computer-aided translation systems are related products which aim to provide human translators with multilingual on-line dictionaries linking word meanings across languages and other similar aids to productivity.

Current commercial MT systems produce as good performance as the best research prototypes. Most perform badly where more than simply grammatical information is essential to translate correctly. For instance, imagine translating *John looked at the cat on the table. It was big.* into French. Mostly, this can be done by pairing word meanings (eg. *cat - chat*, *look at - regarder*, etc.) and knowing about differences of grammar (eg. articles vary with gender – *le chat*, *la table*, etc.), but notice that to translate the second sentence we need to know what *It* refers to because in French this pronoun must agree in gender with its antecedent and this in turn will determine the gender marking of the adjective – *Il/Elle était grand(e).* Working out pronoun antecedents is difficult. Can you see why?
You can try out Alta Vista's version of the Systran MT system at:
`http://babelfish.altavista.com/cgi-bin/translate?`
or the free trial of Translation Experts Ltd, InterTran at:
`http://www.tranexp.com/`

Information Retrieval: Most IR systems for text make little or no use of linguistic knowledge. For example, Web search engines find documents which contain key words specified in the query usually by treating a 'word' as a sequence of characters delimited by white spaces. However, prototype systems exist which are able to provide more focussed access to documents by responding appropriately to queries such as *Find me favourable articles on Clinton* or *Find me articles which claim HIV is not linked to AIDS*. Why can't you achieve this with a conventional search engine? Other systems mix limited MT capabilities with IR to provide search engines capable of dealing with multilingual information sources (such as the Web).

The problem with usefully integrating NLP with IR is that the nature of the language and subject matter to be dealt with is open-ended. Therefore, useful generic systems must deal with unlimited vocabulary and with any grammatical construction found in everyday or specialist language. Superficially, responding appropriately to queries like those above will require full understanding of both the query and the database of texts to be searched. Nevertheless, there are prototype systems which attempt to answer such queries based on quite simple analysis of the text database (though not the query). Can you see how such a system might work in outline?
You can try a demo of an IE system developed by SRI Cambridge at:
`http://www.cam.sri.com/html/highlight_demo.html`
and text summarisation at:
`http://www.mcs.surrey.ac.uk/SystemQ/`

Speech Recognition/Understanding: Commercial speech recognition systems are mostly speaker-dependent continuous word recognisers with vocabularies of a few thousand words. These systems require each user to train the system by saying some sentences a few times and need to be explicitly told which user they are dealing with. They have recognition rates of around 95%. Such systems are now practical for use as 'dictation machines' outputing transcribed text, especially for users who have problems with keyboards. They can be purchased for approximately £100.

Recognition systems model language as a stochastic source and integrate predictions about the most likely next word given the previous two. There are also a few commercial speech understanding systems, but these tend to be limited to isolated word recognition in the context of very restrictive dialogues. For example, telephone-based systems for home banking. Most research is now focussed on extending the success of speech recognition to domains where more flexible dialogue and a greater level of understanding needs to be integrated with speech recognition.

If you have a browser with the right plugins you can look at a video of a speech dialogue system at:
`http://www.cs.rochester.edu/research/trips/movies/`

(Text-to-)Speech Synthesis: Commercial speech synthesisers mostly work from text since this is the most natural form of input for most users. However, systems exist which synthesise higher quality speech from meaning representations and from acoustic representations of a more limited range of utterances. DECTalk is one sophisticated text-to-speech product, producing intelligible speech from more or less unrestricted text, though it has problems with names, acronyms etc. Nevertheless the speech is not natural sounding and requires extra effort to comprehend, making it hard work for large quantities of text; eg. as a book reading machine (perhaps for a blind person).

Research prototype synthesisers have concentrated on improving the quality of synthesis by modelling prosody (intonation, rhythm, pausing) more naturally, incorporating voice quality and dialect differences, and working from less ambiguous representations of utterances than conventional orthography. However, its prob-

ably fair to say that synthesis is not a major focus of current research because existing systems have not turned up lucrative markets. Why do you think that text-to-speech synthesisers are bad at dealing with names, acronyms and intonation?
You can take a look at Bell Labs Text-to-speech system at:
`http://www.bell-labs.com/project/tts/`

3.3 What is Natural Language?

A useful way to think about this question is to consider the differences between natural and formal languages (eg. English and a programming language or a logic). Here are some of the parameters along which they differ – natural vs. invented, range of functions, size of grammar & vocabulary, ambiguity, diversity, ubiquity. However, both human invented and natural languages are strikingly different in comparison to animal 'languages', such as vervet monkey alarm calls, the bee 'honey dance', or chimps taught sign languages, because of their immense productivity. That is, the ability to express an unlimited number of messages (programs or whatever) given a finite vocabulary and finite set of grammatical rules. Can you think of further differences between animal communication systems, human invented languages and human natural language? What difficulties are these likely to cause for NLP?

There are various ways of conceptualising natural language. The one we choose will affect strongly the theories that we develop and the consequent technology. The dominant 'knowledge-based' view (deriving both from Chomskyan linguistic theory and from Artificial Intelligence) is that language is a stringset (well-formed set of strings) generated by a grammar (a finite set of (Post) production (rewrite) rules). Can you think of any problems which might arise if we take this starting point? A related and increasingly fashionable view is that language is best modelled as the output of a stochastic process. Another minority position is that language is a (complex) adaptive system. Each such approach offers different tools and techniques for studying language, theory formalisation, and development of technology. We'll concentrate mostly on the first and dominant view but discuss the alternative potentially progressive perspectives offered by others in connection with its weaknesses.

3.4 Knowledge Sources

All of the tasks / applications discussed above require the automated deployment of various types of linguistic and (often more) general knowledge. The knowledge-based approach to NLP (in common with much work in Artificial Intelligence) attempts to model all (or as much as possible) of this knowledge explicitly, usually by getting 'experts' (linguists) to manually encode it in an appropriate formal and computationally tractable representation. Below I briefly introduce the different types of knowledge required in the applications considered above:

Phonetic knowledge is about the acoustic and articulatory properties of the sounds which can be produced by the human vocal tract, particularly those which are utilised in the sound systems of languages. For example, the sound [b] is a voiced, bilabial plosive; that is a burst of sound is produced by forcing air through a constricted glottis to make the vocal chords vibrate and releasing it from the oral cavity by opening the lips. [p] is the same except that it is unvoiced – the glottis is not constricted and the vocal chords don't vibrate. Say *bun, pun, but, putt* and decide whether [n] and [t] are voiced or unvoiced, by placing a finger gently on your vocal chords. Acoustically, such distinct sounds (mostly) create distinct patterns if we do a spectral analysis of the waveform they produce (measuring time, frequency & intensity).

Phonological knowledge concerns the use of sounds in a particular language. English makes use of about 45 phonemes – contrastive sounds, eg. /p/ and /b/ are contrastive because *pat* and *bat* mean different things. In Vietnamese these sounds are not contrastive. Phonemes are not always pronounced the same eg. the [p] in /pat/ is different from that in /tap/ because the former is aspirated, but aspiration or no aspiration /pat/ still means PAT. Phonology goes beyond phonemes and includes syllable structure, intonation, accent, and so forth. (Note the use of [x] for a sound and /x/ for the related phoneme.)

Morphological knowledge concerns the structure and meaning of words. Some words, such as *send*, appear to be 'atomic' or monomorphemic others, such as *sends, sending, resend* appear to be constructed from several atoms or morphemes. We know these 'bits of words' are morphemes because they crop up a lot in other words too – *thinks, thinking, reprogram, rethink*. There is a syntax to the way morphemes can combine – the affixes mentioned so far all combine with verbs to make verbs, others such as *able* combine with verbs to make adjectives – *programable* – and so forth. Sometimes the meaning of a word is a regular, productive combination of the meanings of its morphemes – *unreprogramability*. Frequently, it isn't or isn't completely eg. *react, establishment*.

Lexical knowledge is about particular idiosyncratic properties of words; eg. what sound or orthography goes with what meaning – *pat* or /pat/ means PAT, irregular morphological forms – *sent* (not *sended*), what part-of-speech a word is, eg. *storm* can be noun or verb, semi-productive meaning extensions and relations, eg. many animal denoting nouns can be used to refer to the edible flesh of the animal (*chicken, haddock* etc) but some can't (easily) *cow, deer, pig* etc., and so forth.

Syntactic knowledge concerns the way in which words can be combined together to form (grammatical) sentences; eg. *revolutionary new ideas appear infrequently* is grammatical in English, *colourless green ideas sleep furiously* is grammatical but nonsensical, whilst **ideas green furiously colourless sleep* is ungrammatical too. (Linguists use asterisks to indicate 'ungrammaticality', or illegality given the rules of a language.) Words combine syntactically in certain orders in a way which mirrors the meaning conveyed; eg. *John loves Mary* means something different from *Mary loves John*. The ambiguity of *John gave her dog biscuits* stems from whether we treat *her* as an independent pronoun and *dog biscuits* as a compound noun or whether we treat *her* as a demonstrative pronoun modifying *dog*. We can illustrate the difference in terms of possible ways of bracketing the sentence – (john (gave (her) (dog biscuits))) vs. (john (gave (her dog) (biscuits))).

Semantic knowledge is about the manner in which lexical meaning is combined morphologically and syntactically to form the meaning of a sentence. Mostly, this is regular, productive and rule-governed; eg. the meaning of *John gave Mary a dog* can be represented as (SOME (X) (DOG X) & (PAST-TIME (GIVE (JOHN, MARY, X)))), but sometimes it is idiomatic as in the meaning of *John kicked the bucket*, which can be (PAST-TIME (DIE (JOHN))). (To make this notation useful we also need to know the meaning of these capitalised words and brackets too.) Because the meaning of a sentence is usually a productive combination of the meaning of its words, syntactic information is important for interpretation – it helps us work out what goes with what – but other information, such as punctuation or intonation, pronoun reference etc can also play a crucial part.

Pragmatic knowledge is about the use of language in context, where context includes both the linguistic and situational context of an utterance; eg. if I say *Draw the curtains* in a situation where the curtains are open this is likely to be a command to someone present to shut the curtains (and vice versa if they are closed). Not all commands are grammatically in imperative mood; eg. *Could you pass the salt?* is grammatically a question but is likely to be interpreted as a (polite) command or request in most situations. Pragmatic knowledge is also important in determining the referents of pronouns, and filling in missing (elliptical) information in dialogues; eg. *Bill always gives his wife his wages. Bob does so too.*

General Knowledge also plays an important role in language interpretation; for example, if I say *Lovely day* to you whilst we are both being soaked by heavy rain, you will use knowledge that people don't usually like rain to infer that I am being ironic. Similarly, the referents of names and definite descriptions, if not determined situationally, are determined through general knowledge which may be widely shared or not; eg. *the prime minister*, *Bill, my friend with red hair.* Pronoun reference can also often only be determined using general knowledge; eg. *John looked at the cat on the table. It was furry / white / varnished / fat / china / frisky ....*

3.5 Knowledge-based vs. Statistical Approaches

Whilst, knowledge-based approaches require significant manual encoding of the knowledge sources relevant to a specific NLP application, statistical approaches do not attempt to model explicitly (much of) the linguistic and other knowledge needed to make NLP systems work. Instead they attempt to acquire this information automatically from large quantities of spoken or written material (corpora) and represent it implicitly in the form of probabilities. One successful example of such an approach is the word prediction components of most continuous word recognition systems (briefly described above), which incorporate statistics concerning the frequency of 3 word sequences (trigrams) derived from a large corpus; given 2 word contexts it then uses these statistics to predict the n most probable next word candidates and this is used in conjunction with acoustic information to select an appropriate sounding and contextually probable word. For example what words do you think might follow the sequences *married a*, *What is*, or *share prices*? Can you see any of the limitations of this technique? Another example is part-of-speech tagging – assigning the appropriate part-of-speech to a word in context on the basis of the probability of trigram sequences of particular parts-of-speech in text and of the probabilities with which particular words take specific parts-of-speech.

Statistical approaches are less radical a departure from the knowledge-based approach than is sometimes claimed. Typically, a statistical system is based on a *language model*, a stochastic generator of a language (such as a Markov model, i.e. a probabilistic finite-state machine) which can be viewed as a specification of a grammar within the knowledge-based approach. However, typically such 'grammars' are much more liberal, preferring to treat the 'ungrammatical' as highly improbable, as well as resolving ambiguity and indeterminacy on the basis of the differential frequency of linguistic phenomena in a training corpus, and usually language models are coarser descriptions of language than those developed in the knowledge-based framework.

3.6 Text Understanding

This course will mostly focus on text understanding as a kind of generic or core application of NLP. We can think of text understanding as a mapping (or transduction) between two levels of representation – orthography and meaning. The representation of orthography is straightforward enough, but what is a representation of meaning? There are many answers to this question, but we will assume that the language that we use to represent meaning must be unambiguous and itself have a formal semantic interpretation (model theoretic or proof theoretic or both); that is, it must be a logic (of some kind). Furthermore, for NLP we would like this logic to be amenable to theorem proving techniques, because we want to be able to use automated deduction to capture the inferences necessary to interpret language.

This approach might seem too restrictive. However, rather than defend it by positive argument at this stage, I will make use of a 'what else?' argument: suppose that I tell you that the meaning of *John loves Mary* is *LOVE(JOHN34, MARY29), this is really no different from my telling you that the translation of this sentence into Mumbo-Jumbo – a language spoken by the remnants of an obscure tribe, which I can assume with a fair degree of confidence that none of you speak – is *Mbarim huoki Xhoni*. On the other hand, if I tell you that *LOVE(JOHN34, MARY29) is a formula of first-order predicate logic (FOL) in which *LOVE is a 2-place predicate representing a relation between two human beings (which we call *love* in English, *aimer* in French, etc.) and that JOHN34 and MARY29 are constants denoting two unique individuals (eg. John Lennon and Mary Whitehouse) you are in a better position to evaluate the claim that I'm making. Can you see any problems with this analysis? Can you give a better account of meaning (in this sentence)? We'll return to the use of logic as a representation of meaning in natural language later.

Text understanding can be divided into two components: decoding, a language-specific and medium-specific process, and inference, a language/medium-independent refinement of the resulting interpretation. For example, the two utterances: *Lovely day!* and *Quelle belle journée!* might be decoded into an initial representation something like: SOME(X) & BEAUTIFUL(X) & DAY(X). However, the inference that the speaker is being ironic about today proceeds from comparing this initial representation with what is known about the (extra)linguistic context of utterance. NLP, and this course, are primarily concerned with the decoding (and encoding) problem, but cannot ignore the inference component since it is strongly intertwined with the recovery of meaning. For example, is resolving the anaphoric referent of the pronoun *he* in the following sentence a matter of decoding or inference? – *Every Cambridge student thinks he is the brightest person on the planet* (Stephen Hawking or you?!).

4 Words

4.1 Parts-of-speech

Words can be analysed into parts-of-speech (PoS) – major lexical syntactic categories, such as N(oun), V(erb), A(djective), P(reposition), or more minor categories, such as Comp(lementizer), Det(erminer), Deg(ree intensifier), and so forth:

```
N: car, cars; woman, women...
V: thinks, thinking; sold, selling...
A: old, older, oldest; pedantic...

P: in, on, with(out), although...
Comp: that, if...
Det: the, a, those, that, some...
Deg: so, very...
```

N,V,A are the categories of the contentful or open-class vocabulary. Membership of these categories is large (as a glance at any dictionary will tell you) and open-ended (people invent new words (neologisms) like *fax*, *biro*) and often open-class words belong to more than one category (e.g. *storm* can be a noun or verb, and morphologically-related *stormy* is an adjective); that is, they are ambiguous in terms of lexical syntactic category. (Some words are ambiguous at the level of lexical semantics though not in terms of lexical syntactic category e.g. *lamb*, N: animal, meat.)

The other categories are those of functional or closed-class words, which typically play a more 'grammatical' and less 'meaningful' role in a sentence. Membership of these categories is smaller and changes infrequently. For example, prepositions convey some meaning but often this meaning would be indicated by case endings or inflection on words in other languages and sometimes there are English paraphrases which dispense with the preposition: *Kim gave a cat to Sandy / Kim gave Sandy a cat*. Degree intensifiers in adjectival or adverbial phrases *very beautiful(ly)* convey a meaning closely related to the comparative suffix *more beautiful / taller*. Determiners, such as the (in)definite articles (*the, a*), demonstrative pronouns (e.g. *this, that*) or quantifiers (e.g. *some, all*) help determine the reference of a noun (phrase) – quite frequently articles are absent or indicated morphologically in other languages (hence the common non-native speaker error of the form *please, where is train station?*).

The complete set of lexical syntactic categories (for English) depends on the syntactic theory, but the smallest sets contain around 20 categories (almost corrsponding to traditional parts-of-speech) and the largest nearly a thousand.

Often words are ambiguous between different lexical categories. What are the possibilities for *broken*, *purchase*, *that* and *can*? There are diagnostics for determining the category appropriate for a given word in context, e.g.s: if a word follows a determiner, it is a noun: *the song was a hit*; if a word precedes a noun, is not a determiner and modifies the nouns meaning, it is an adjective: *the smiling boy laughed* – can you think of an exception to the last rule?; a verb can follow a noun; and an auxiliary verb can precede a main verb. It is possible to resolve most ambiguities over lexical syntactic category just by looking at the immediate syntactic context (i.e. the category of the preceding and following word). Consider, for example, *I can can a can*. A dictionary or lexicon might tell us that *can* is ambiguous between

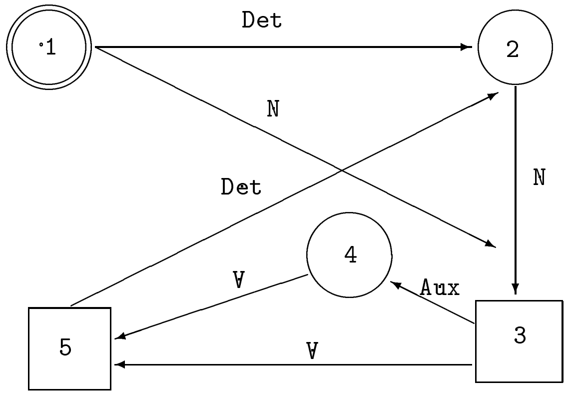


Figure 1: Simple FSA for PoS Tagging

'modal auxiliary verb' (Aux), 'main verb' (V) and 'noun' (N), so we start out with the possibilities in a):

```
a)
I can can a   can
N N   N   Det N
  Aux Aux     Aux
  V   V       V

b)
I can can a   can
N Aux V   Det N
```

We can obtain the correct disambiguated sequence in b) by applying rules like those above to a) to remove contextually incompatible categories. This is the basis for part-of-speech tagging using probabilistic finite-state automata (FSAs, or more-or-less equivalently (hidden) Markov Models, (H)MMs). We will sketch how such rules might be represented as a non-deterministic FSA and how this can be used to resolve such ambiguities. We represent a FSA as a transition diagram consisting of a start state (double circle, labelled state 1), two possible finish states (squares, 3, 5), and some intermediate states. We draw directed arcs (arrows) representing transitions between states derived by inferring legal sequences of pairs of categories from the diagnostic rules like those above (Det-N, N-V, N-Aux, Aux-V, etc.). The FSA generates or accepts strings consisting of sequences of PoS symbols defined in terms of allowable transitions between the start and end states. See Figure 1. Now finding the correct sequence of categories b) from a) corresponds to finding a sequence of PoS symbols which can be accepted by the FSA in Figure 1. The correct and unique state sequence is 1-3-4-5-2-3. In general, of course, we are not guaranteed to get a single answer so probabilistic FSAs return the most likely one. For example, if we wanted our PoS tagger to cover an example like *The water meter is broken* where *water meter* is a noun-noun sequence or noun compound, we would need to add an arc accepting N and allowing iteration in state 2 so the sequence Det-N-N-Aux-V is accepted. However, once we do this we will no longer get a single answer from the lexical entries for *I can can a can* with this new FSA. Can you see why?

Suppose we want to return the most likely path from the set found. One way of doing this is to associate probabilities with transitions in the network representing the conditional probability that one PoS tag will be followed by another, $P(tag_k \mid tag_i)$, and that making a specific transition associated with a PoS tag the FSA will output a specific word, $P(word_j \mid tag_k)$. In order to do this correctly, we must ensure that the probabilities of all transitions from a given state sum to one and that the probabilities of all words given a PoS tag sum to one. Then we have defined a *language model* or stochastic 'hidden' language generator in which the probability of all the sentences generated by the model is one. Can you explain why this is true?

Finding the most probable path or state sequence requires taking the product of the lexical and transition probabilities for each possible path through the FSA, so if we have an observed sequence of words like *I can can a can* we find all the sequences of 'hidden' PoS symbols that could have generated that sentence and compute the probability of each such sequence: $P(NP \mid START) \times P(I \mid NP) \times P(Aux \mid NP) \times P(can \mid Aux) \ldots$ In general, there will be an exponential number of such paths bounded by the length of the sentence and the size of the FSA. However, there is a method of incrementally pruning submaximal paths which guarantees finding the maixmal one in linear time. Can you see how this might work?

These transition probabilities and lexical probabilities can be estimated from a corpus which has been manually annotated with the correct PoS symbols. A standard method of parameter estimation is to maximise the probability of the training corpus relative to the model adopted. In fact it is possible to adopt the maximally uninformative 'ergodic' HMM in which any state is reachable from any other state and thus any sequence of PoSs is accepted. Can you draw an ergodic version of the FSA in Figure 1? The parameters of an ergodic model for a given set of PoS symbols can be estimated from the training data so that the probability of each sentence of the training data is as high as possible, and thus the probability of the training data is maximised. The probability of a sentence, $P(S)$, is the sum of the probabilities of the different paths through the HMM which could have generated the sentence. This probability will be maximised if the transition probabilities (TP) and lexical probabilities (LP) are estimated by their relative frequency in the annotated training data; e.g:

LP $= P(can \mid Aux) = \frac{freq(can{:}Aux)}{freq(Aux)}$

TP $= P(Aux \mid NP) = \frac{freq(NP+Aux)}{freq(NP)}$

Given 1M+ words of training data such systems perform with about 95% accuracy, defined as the ratio of correct PoS symbols assigned to all symbols assigned on 'similar' test material. It turns out that lexical probabilities alone yield 90% accuracy – the fact that *can* is most frequently a modal auxiliary is really the most important piece of information in our toy example. However, lexical information is the most difficult to acquire accurately and comprehensively. Why is this the case? What might we do to PoS tag a sentence containing a word (form) or requiring a word:PoS pair that didn't occur in the training data? Maximum likelihood estimation will assign zero probability to such events. Can you see why? (We'll return to the problem of unseen events when we consider probabilistic approaches to parsing.)

You can read more about FSAs in Rayward-Smith, and their uses in NLP in Gazdar & Mellish, Allen, Charniak or Manning and Schutze, ch9. De Rose, 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics*, 14.1 pp 31–39, 1988 is a good introductory paper on PoS disambiguation or 'tagging' using probabilistic FSAs which includes an algorithm for optimising the search for the most probable path, though here lexical probabilities represent $P(tag_k \mid word_j)$ so the FSA does not define a language model.

You can try out HMM PoS taggers for English and other languages at:
`http://www.xrce.xerox.com/research/mltt/Tools/pos.html`

4.2 Morphology

Words are rarely atoms (monomorphemic). They have structure that requires analysis before they can be looked-up in a dictionary / lexicon and sometimes before they can be interpreted (*great-great-great grandmother*, *anti-anti-missile*). Inflectional morphology deals with variants which don't change part-of-speech and are determined by syntactic context; derivational morphology with new word (sense) formation rules which usually do change part-of-speech.

```
change, changes, changing, changer, changers, changeable
V       V[3rd,sg] V[prog]  N        N[pl]     Adj
N[sg]   N[pl]

un  re program able ity
Adj V  V       Adj  N

((un ((re program) able)) ity)
'the-property-of not being-able to-program (x) again'
```

Combining free and bound morphs (stems and affixes) usually involves spelling changes – *able* → *ability*, *change* → *changing*. Each morph is labelled with its lexical syntactic category and the rightmost morph (usually) gives the category of the complete word. There can be ambiguity in decomposing words:

```
formally   form  ally    for  mal  ly    form  al  ly    form  all  y
           N/V   N/V     P    A/N  Adv   N/V   A/N Adv    N/V  Det  Adj
```

so algorithms need to use heuristics (longest morphemes), statistics, and/or be guided by a lexicon. Sometimes morphemes do not combine productively to form meanings which are rule-governed: *establishment* 'the setting up of something' vs. 'the ruling elite'.

English is relatively isolating (not much inflectional morphology), languages like Hungarian, Finnish and Turkish have many variants (often 100s sometimes 1000s) of each verb. Other like Arabic use infixation rather than suffixation (prefixation): *ktb*, *katab* etc.

4.3 Finite-state Transducers and Spelling Changes

Finite-state transducers (FST) can be used to apply spelling change rules. A FST is a FSA with an output tape. Each transition in the machine specifies an input:output pairing of symbols or variables over symbols. Transduces between the surface form and lexical form for a morphologically complex word (surface:lexical). For example, *i* to *y* as in *flies* to *fly+s*: '=' means 'any other character', '+' is the morpheme boundary character, 0 is the null character. 1 is the Start/Finish state. 6 is a finish state. Assume the lexicon is:

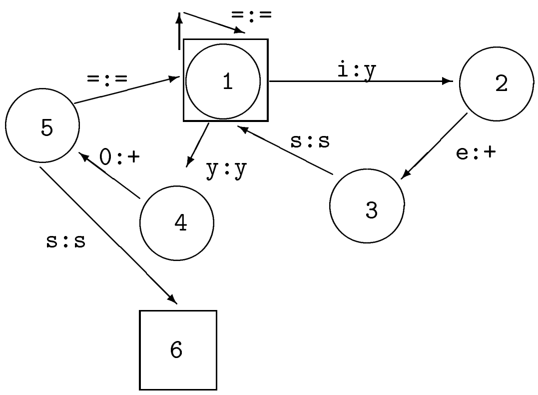


Figure 2: FST for Spelling Changes

```
fly
+s
+ing
```

Running the FST in Figure 2 on *flies*, *fly0s* and *flying* yields the following sequence of transitions and output tape:

```
Surface  f l i e s
State    1 1 2 3 1
Lexical  f l y + s

Surface  f l y 0 s
State    1 1 4 5 6
Lexical  f l y + s

Surface f l y 0 i n g
State   1 1 4 5 1 1 1
Lexicon f l y + i n g
```

How could you modify the FST so that *flys* resulted in an analysis which terminated at the finish state rather than blocking? Do you think the FST is reversible? – could we generate words of English from the lexical forms?

The lexicon can be represented as a character tree (ie. a deterministic FSA with null transitions from final states to the start/finish state.

This approach can be extended to all spelling rules by 1) writing a FST for each rule, 2) writing a 'default' FST which says that surface:lexical symbols are identical unless a rule says otherwise, 3) null characters at a particular point must be accepted by all transducers at that point. You can read more about FSTs in Rayward-Smith and their uses in NLP in Gazdar and Mellish. For more on the composition of spelling rule FSTs and associated rule formalisms see Kaplan and Kay 'Regular models of phonological rule systems', *Computational Linguistics* 20.3, 1994, 331–378.)

4.4 Semi-productivity

The meaning of a complex word can be built up using the same techniques that we will look at in the next three lectures. However, many lexical rules are only semi-productive. Consider the following which could be analysed as formed by suffixation of *+er* to a verb stem:

```
teach/teacher, buy/buyer, smoke/smoker   (agentive)
dry/dryer, freeze/freezer  (instrumental)
stick/sticker  (result)
*station/stationer (newsagent), *lie/lyer (liar)
?steal/stealer (thief) (only) a stealer of Porsches (synonymy)
?hammer/hammerer (lexical form)
```

Blocking, preemption by synonymy or by lexical form, is a big source of semi-productivity – avoiding unnecessary redundancy/ambiguity in the lexicon. One possible solution is to collect statistics on the frequency of words but this is complicated by the need to collect statistics on word **sense** usage (cf *stationer*). How could we use these statistics (assuming we can gather them) in conjunction with rules and lexical entries to achieve the desired effect? (See Manning and Schutze, ch7 for more on statistics and word senses.)

4.5 Summary

We have looked at some of the problems in getting from textual input to initial information about the words comprising a text by looking them up in an electronic dictionary. Even with this apparently straightforward task, we can see some of the tension between the knowledge-based and statistical view of NLP. Though we have assumed the existence of a dictionary and rule systems for morphology, part-of-speech selection in context and so forth, the high degree of ambiguity in natural language makes determinate application of these rules systems impossible. However, there are other questions too: How large will such rule systems and dictionaries need to be to 'cover the language'? For example, how many headwords would you need in your dictionary to reliably achieve 99% coverage of an arbitrary document found on the Web?

5 Syntax

In this section, we will consider syntactic knowledge in more detail and introduce several formalisms in which to represent syntactic information, evaluate their adequacy from various perspectives, and finish by considering what the syntax of NLs in general is like, as opposed to the syntax of English in particular.

5.1 Why do Languages have Syntax?

In order to gain further insight into the function of syntax, consider what a language without syntax would be like. Such a language would be just a vocabulary and a sentence would be any set of words from that vocabulary. Now imagine that this language has English as its vocabulary. A "sentence" in this imaginary language is shown below:

```
      the         hit(s)          a

           with           tramp(s)

  sharp            poor     rock(s)     some

           boys          cruel
```

There is no clue which words should be interpreted with which others in this sentence, so there are many possible interpretations which can be "translated" into real English, as in (1a,b).

(1) a The cruel boy(s) hit(s) some poor tramp(s) with a sharp rock.
    b The cruel, sharp tramp with a rock hit some poor boys.

How many more possible interpretations can you find? Without syntax, sentences would be very ambiguous indeed and, although the context of utterance might resolve some of these ambiguities in everyday communication, imagine trying to discuss politics, philosophy or to explain the design of a computer in such a language!

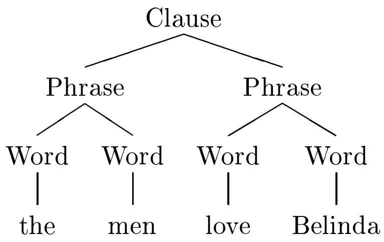
Genie (the alias for a girl who was not exposed to language until she was 13 years old) provides us with a glimpse of what life would be like without syntax. Some of Genie's utterances are given in (2).

(2) a Spot chew glove
    b Tummy water drink
    c See Mama Friday
    d Sick people lady drive ambulance
    e Genie bad cold live father house
    f Father house live father Genie bad cold

Word order is fairly random; for instance, the verb turns up at the beginning, end and in various positions in the middle of these utterances. Inflectional morphology is missing and the overall effect is like a badly constructed telegram. In most cases it is possible to guess, the meaning intended, but as the sentences get longer this gets more difficult. See how many interpretations you can find for the last two examples. Genie's story and language is discussed by Curtiss in Obler & Menn (1982) *Exceptional Language and Linguistics.* Academic Press.

5.2 Constituency

Words go together to form syntactic units of various kinds called constituents. We will recognise the following types of constituents: words, phrases and clauses. These constituents form a hierarchy:



Words, phrases and clauses can be of different types depending on their constituents. Constituents tend to represent coherent units of meaning in some sense. For example, *The men* seems to be interpretable independently of the rest of this sentence. However, what exactly is meant by coherent unit of meaning is not clear, so we will try to find more precise ways of defining constituents.

5.3 Lexical Categories

Traditionally, words are categorised according to parts-of-speech. More recently, parts-of-speech have been absorbed into the more general concept of a syntactic category. The major lexical categories are noun, verb, preposition and adjective. There are a variety of minor categories, such as the determiners, intensifiers, complementisers, and so forth. (Adverbs are usually treated as adjectives of some kind - we will ignore them for now.) We will adopt abbreviations for these categories, as below:

```
N  (eg. boy, castle, storm)

V  (eg. kiss, hit, storm)

A (eg. pretty, many, old)

P (eg. to, of, without)

Det (eg. the, a)

Comp (eg. that)
```

5.4 Phrasal Categories

Each of the major lexical categories is associated with a corresponding (noun, verb, adjectival or prepositional) phrase in which the major lexical category is obligatory, as illustrated:

```
NP (eg. boys, the boy, an old castle, kings of England)

VP (eg. run, kiss Belinda, give me a present)

AP (eg. old, very old, quite pretty, difficult to understand)

PP (eg. up, to the house, without me)
```

Can you think of sentences which contain these constituents? Where do these constituents occur in relation to each other? If we look at the noun phrase (NP) in a bit more detail, we can see that the head noun is the only obligatory element of every NP and that this noun can co-occur with Det(erminers), APs and PPs, as illustrated in (3).

(3) a The castle is old
 b *The is old
 c The big castle is old
 d *The big is old
 e The castle by the hill is old
 f *The by the hill is old
 g Castles are interesting

5.5 Clausal Categories

Clauses can be independent sentences or embedded inside other sentences. There are various types:

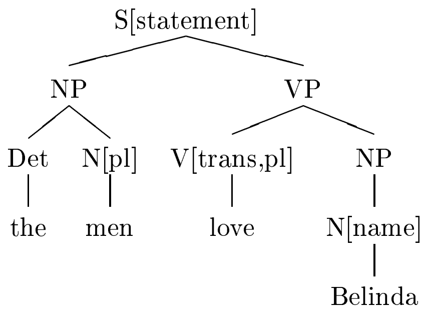
```
S[statement] (eg. They kiss Belinda, God exists)

S[question]  (eg. Did he kiss Belinda, Who did he kiss)

S[command] (eg. kiss Belinda, get up)

S[relative] (eg. who he kissed, who likes me)

S[complement] (eg. that Max kissed Belinda)
```

A couple of examples of sentences with further clauses embedded inside them are given in (4). Can you recognise which types of clause they are?

(4) a John thinks that castles are interesting.
 b John likes the man who he met yesterday.

5.6 Phrase Marker Trees

The constituency of a particular sentence can be shown in a phrase marker tree. For instance, we can now show the types of the constituents for *The men love Belinda*:



So far we have seen examples of various constituents but no definition of what a constituent is or how to know when a sequence of words is a constituent.

5.7 Diagnostics for Constituency

There are various tests for constituenthood none of which are foolproof, so we will refer to them as diagnostics. Most rely on our intuitions about what is and isn't a grammatical sentence. This is an acquired and quite sophisticated intuition about the correctness of a sentence considered in isolation. For example (5)

(5) *Reagan thinks bananas

is not a complete grammatical sentence considered independently. Therefore, as syntacticians, we would reject it (hence the asterisk). However, it would be quite possible for this sequence of words to occur in a conversation; (6)

(6) a What kind of fruit does Bush like?
 b Reagan thinks bananas

In this context, the missing constituents are "understood" and the sequence is perfectly acceptable as an elliptical form of (7).

(7) Reagan thinks that Bush likes bananas.

The most important diagnostic is the possibility of substitution or replacement of a possible constituent by another form (particularly a proform, such as *they*, *that*, *do so*, and so forth). If the replacement can be made without altering the grammaticality of the sentence, then this suggests that the replaced words form a constituent of the same type as those which replaced them. For example, the NP *The men* can be replaced by a wide variety of material:

```
The men                            love  Belinda
They
*He
Some friends of hers
The men who she met
*The old woman with grey hair
*the
*quick
*hit
*with the man
```

This shows us that all of the unasterisked sequences can be NPs. The same technique can be used to work out what words are (transitive) verbs:

```
The men loved         Belinda
        liked
        hit
        chased
        talked to
        looked at
        *gave
        *likes
        *think
        *pretty
        *girl
```

What is a transitive verb? What makes it different from other types of verb, such as *give* or *think*?

Constituents, but not partial constituents can be moved around in a sentence, for example (8).

(8) a The old man has come to dinner.
    b Has the old man come to dinner?
    c *The has old man come to dinner

In this case *the old man* is a NP and *has* is an auxiliary verb, but *old man* is only part of the NP. So movement is also a diagnostic for constituency.

Parentheticals and extra material can be inserted between some phrasal constituents but not within them; for example (9).

(9) a The President of America, Ronald Reagan, is over 70.
    b *The President, Ronald Reagan, of America is over 70.
    c *The President of America is, Ronald Reagan, over 70.

The omissibility of a potential constituent, either because of its optionality or because it is "understood" in some context, is a sign of constituenthood. For example, the PP *of the old man* can be omitted from (10).

(10) Some friends of the old man came to dinner.

but *it of the old*, which is not a constituent in this sentence, cannot.

If two sequences can be coordinated with a conjunction (eg. *and*), they may be constituents of the same category, as illustrated in (11)

(11) a Max and Belinda kissed each other
     b The old men and women came to dinner
     c The old man and his young nephew came to dinner
     d Max and Belinda divorced and remarried each other
     e Kim kissed Sandy and remarried her
     f That rather old and very unreliable car belongs to Max
     g Kim washed up and Sandy watched the TV

Can you name the constituents coordinated in each case?

5.8 Generative Grammar

Generative linguists try to develop explicit rules for analysing sentences. Below illustrates some of these rules for English:

a) S[statement] $\rightarrow$ NP VP
b) NP $\rightarrow$ Det N
c) NP $\rightarrow$ N[name]
d) VP $\rightarrow$ V[trans] NP

These rules will assign the sentence *The men love Belinda* the same analysis and phrase marker tree that was proposed above, repeated as a labelled bracketing below. (Can you see how?)

```
S(NP((Det The) (N[pl] men)) VP((V[trans] kiss) NP(N[name Belinda)))
```

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

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. (Note eg. the lack of brackets round features like 'name' which pick out subcategories of nouns.)

CF rules encode (immediate) dominance and precedence relations between (non-)terminal categories of the grammar. All grammars have a designated root or start symbol (see e.g. Rayword-Smith or Gazdar and Mellish for more formal and detailed formal definitions of CFGs). To make the grammar complete, we also need a dictionary (often called a lexicon) in which we pair words with their lexical categories. Grammar 1 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.              Hannah : 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.
```

What sentences does this grammar generate? Is this set of sentences finite? Are they all grammatical?

5.9 Derivations

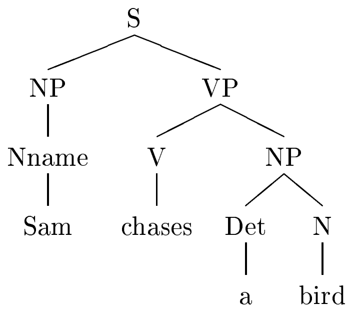
Given a CF PSG and lexicon, we can determine whether a sentence is or is not generated by attempting to construct a derivation (tree) for it. To construct a leftmost derivation, we start with the root symbol of the grammar and always rewrite (expand) the leftmost non-terminal category in the current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



5.10 Ambiguity

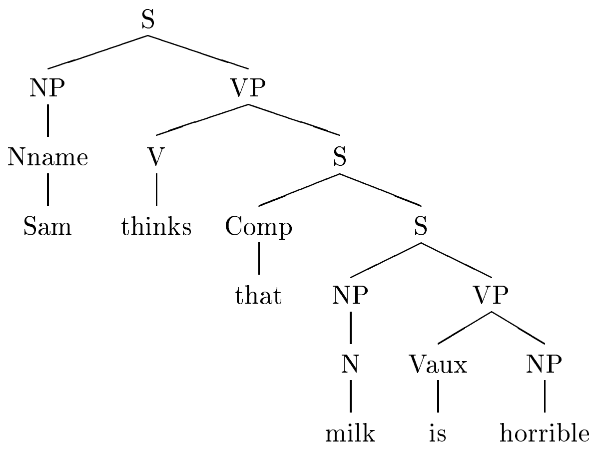
A sentence is said to be ‘ambiguous’ when it can be assigned two (or more) distinct semantic interpretations. A distinction is often made between two types of ambiguity, ‘structural’ and ‘lexical’. Broadly speaking, a sentence will be said to be structurally ambiguous if a syntactic analysis of it results 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. (12a) is an example of structural ambiguity and (11b) an example of lexical ambiguity.

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

In (12a) the word *that* is ambiguous between a determiner reading and a complementizer reading. A complementizer is a word that introduces a dependent clause. The most common complementizer is *that*, but other examples include *for*, *whether* and *if*, as in (13).

(13) a Sam wonders whether Felix is in the garden
     b Hannah would prefer for Felix to be in the garden
     c 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 complementizer it forms a constituent with the clause *milk is horrible*. The tree below shows the complementizer analysis where it is all milk that Sam considers to be horrible.



Can you construct the tree for the other reading? With (12b) we will end up with the same phrase marker tree but different senses of the noun *trees*. Can you construct it and add the rules and words to grammar 1 needed to generate these examples, ensuring that (12a) is assigned two analyses corresponding to the structural ambiguity? 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.

5.11 Descriptive Adequacy of CF PSG

CF PSGs have some strengths as a theory of generative grammar; for instance, they seem to account for constituency quite well and by using recursion we can capture the syntactic productivity of language. There are infinitely many grammatical sentences of English; consider, for example, the sentence in (14).

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

We can introduce the recursive CF rule below

NP $\rightarrow$ NP PP

which will assign a phrase marker tree to this example which corresponds to the reading in which the telescope is mounted on a monument to be found in the park where the man is. Draw the tree by adding this rule to grammar 1 along with some appropriate lexical entries. What rule(s) would we need to add if we wanted to capture the reading in which Sam saw the man whilst Sam was in the park using the telescope mounted on the monument? As a clue, assume that PPs which modify actions or events attach to the VP node.

Nevertheless if you have explored Grammar 1, 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 (15).

(15) 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 subject-verb agreement in our grammar to block examples like (16)

(16) 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 grammar 1, we will end up with the much larger CF PSG, Grammar 2.

```
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 NP and V(P) in an S agree in number and person'

5.12 Unification-based Generative Grammar

CFGs utilise atomic symbols which match if they are identical, unification-based phrase structure grammars (UB PSGs) utilise complex categories which match by term unification (as in Prolog). They provide us with a means to express the number/person 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]

As well as being able to specify values for features we will also allow features to take variable values which can be bound within a unification-based PS rule. The rules below express num/per agreement:

s $\rightarrow$ np:[num=N, per=P] vp:[num=N, per=P]
vp:[num=N, per=P] $\rightarrow$ v:[num=N, per=P] np:[]

(From now on, we'll use a notation close to Prolog in which category names and feature attributes and values will be in lower case, whilst feature variable values will be single upper case letters when discussing UBGs.)

Mostly, words in the lexicon will have fully specified categories but categories in rules often contain variable values, so they generalise across subcategories. Fixed-arity term unification will be used as the operation to match categories during the construction of a phrase structure; that is, two categories will match if they contain the same set of feature names and for any given feature name they do not have distinct values. The resultant category, if unification succeeds, is the one obtained by 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=1, num=sg] | np:[per=1, num=sg]
    |                     |                    |
e.  |      N:[]           |        N:[]        |    N:[]
    |                     |                    |
f.  | v:[subc=intrans,    | v:[subc=intrans,   | v:[subc=intrans,
    |    per=3, num=sg]   |    per=P, num=N]   |    per=3, num=sg]
g.  | vp:[in=F, out=G]    | vp:[in=F, out=F]   | vp:[in=H, out=H]
h.  | vp:[in=1, out=2]    | vp:[in=F, out=F]   | fail
```

Unification can be defined in terms of subsumption:
If two categories (roughly Prolog terms), A and B, unify, this yields a new category, C, defined as the smallest (most general) category subsumed by A and B, otherwise fail.

A category consists of a category name (the functor) and a set of features. Features are made up of attributes (eg. per) and values/variables (eg. 1 or P).

Category A subsumes category B (i.e. B is more specific than A) iff:
1) every attribute in A is in B (and every attrbute in B is in A);
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.

(More formal and/or detailed definitions can be found in Prolog textbooks, and Gazdar and Mellish.)

Here follows 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 subcategorisation constraints.

```
Grammar 3

s:[] --> np:[per=P, num=N] vp:[per=P, num=N]
vp:[per=P, num=N] --> v:[per=P, num=N, subc=intrans]
vp:[per=P, num=N] --> v:[per=P, num=N, subc=trans] np:[]
vp:[per=P, num=N] --> v:[per=P, num=N, subc=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, num=N, 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, num=N, pronom=no]
laughs v:[per=3, num=sg, subc=intrans]
laugh v:[per=1, num=sg, subc=intrans]
laugh v:[per=2, num=sg, subc=intrans]
laugh v:[per=P, num=pl, subc=intrans]
chases v:[per=3, num=sg, subc=trans]
chase v:[per=1, num=sg, subc=trans]
chase v:[per=2, num=sg, subc=trans]
chase v:[per=P, num=pl, subc=trans]
gives v:[per=3, num=sg, subc=ditrans]
give v:[per=1, num=sg, subc=ditrans]
give v:[per=2, num=sg, subc=ditrans]
give v:[per=P, num=pl, subc=ditrans]
the Det:[]
a Det:[num=sg]
those Det:[num=pl]
```

Try adding the analogues of some of the other rules and/or lexical entries we developed for the CF PSG, Grammar 1 to the UB PSG, Grammar 3.

5.13 NL and the Chomsky Hierarchy

Our UB PSG grammar for the fragment of English we have covered is CF equivalent in the sense that we could automatically compile it into a finite set of CF rules. However, there is strong evidence that NLs in general are not CF and some evidence that certain (rarish) constructions of English aren't either.

$a^n b^n$ is CF (i.e. ab, aabb, aaabbb, *abb, etc.)

S → a b
S → a S b

A language containing nested dependencies.

$a^n b^n c^n$ is not CF (i.e. abc, aabbcc, *abbcc, etc.) Try to write a CF grammar for it! – a language of cross-serial dependencies. This is an indexed language – between CF and context-sensitive on the Chomsky hierarchy. An indexed grammar can be constructed out of a CF grammar 'backbone' with categories employing at least one recursive feature (i.e. the category set is no longer finitely specifiable).

UB PSG for $a^n b^n c^n$:

s:[s=H|T] → s1:[s=H|T] s2:[s=H|T] s3:[s=H|T]
s1:[s=H|T] → a s1:[s=T]
s1:[s=[]] → a

(and ditto for s2 and s3, where value of s=h|t is a (Prolog) list consisting of a head element and a tail.)

Gazdar & Mellish give another indexed grammar for $a^n b^n c^n$ and discuss these issues in more detail.

Cross-serial dependencies occur in Swiss German of form NP^n VP^n:

mer d' chind em hans es huus lond halfe aastriiche
we the children-ACC Hans-DAT the house-ACC let help paint
we let the children help Hans paint the house

```
 ---------------------------------
|                                 |
mer  d' chind  em hans  es huus  lond  halfe  aastriiche
NP   NP[acc]   NP[dat]  NP[acc]  Vacc  Vdat   Vacc
     |         |        |           |     |      |
      ------------------------------      |      |
               |        |                 |      |
                --------------------------       |
                        |                        |
                         ------------------------
```

So-called MIX languages in which dependencies are arbitrarily intersecting don't seem to occur in natural language.

On the other hand, the evidence that NLs are not regular languages and thus captured by FSAs / Regular grammars is based on the grammaticality of centre/self-embedding constructions like (17)

(17) a The students the police arrested laughed
b ??The students the reporters the police arrested interviewed laughed

(Linguists use ?? to indicate sentences that are unacceptable / very marginal.) These examples require non-regular rules of the form NP $\rightarrow$ NP Srel. However, the depth to which such embedding is grammatical is clearly questionable. The standard story is that they are grammatical to any depth but only acceptable in use to depth 1, or occasionally 2, due to human working memory limitations.

So UB PSGs have the expressive power to capture (all?) rules of NL syntax, but we have also seen that FSAs and CFGs are potentially useful for NLP. (See Rayward-Smith for more detail on the Chomsky Hierarchy / formal language theory.)

6 Parsing

6.1 Recognition vs. Parsing

A recognizer given an input and a grammar decides whether that input is generated by the grammar or not – it returns yes/no.

A parser given an input and a grammar returns one or more phrase marker trees for the input according to the grammar, if the input is generated by the grammar (and an error message or partial parse otherwise). To be 'correct' with respect to a grammar, a parser should return all and only the analyses associated by the grammar with a sentence.

Are either of these computable for natural language? If NL is CF, yes, if NL were at 'top' of Chomsky hierarchy, maybe not. How efficiently?

Decision procedure – recogniser / parser guaranteed to terminate in finite number of steps

Parsing complexity – number of steps to return structures as function of length of input (in words n) – linear eg. 3n, polynomial eg. n^3 exponential eg. 3^n

6.2 Local & Global Ambiguity

Parsing is a procedure which involves applying grammar rules to an input in some sequence.

Left-to-right / Right-to-left – parsing proceeds (usually) left-to-right (much as people – *look out there's a fall...*); this means that the context for a parsing decision does not consist of the entire input.

Parsers not only need to deal with global syntactic ambiguity (more than one analysis), but also local syntactic ambiguity (temporary indeterminacies created by limited nature of the parsing 'window')

Visiting aunts can be boring
The farmer killed the duckling in the barn

Who does ...
Who does Mary want ...
Who does Mary want to kiss ...
Who does Mary want to kiss (Bill / 0)

The cotton clothing is made of grows in Mississipi
Without her contributions to the fund would be inadequate
They told the girl the boy seduced the story

These latter examples are known as 'garden paths' in the psychological literature. They raise the issue of whether humans parse syntactically independently of other information (eg. intonation, semantics, etc.). Can you draw partial phrase marker trees which show the two different syntactic analyses up to the point of the resolution of the ambiguity in (some of) these examples?

6.3 Parsing Algorithms

Top-down / Bottom-up – parsing can proceed top-down, hypothesise an S node and try to find rules to connect it to the input, or bottom-up, starting from the first (or last) word of the input try to find rules which successively combine constituents until you find one S.

Search & Non-determinism – parsing involves search at a number of levels, successively search grammar for a rule containing a category which matches the current item, search for combinations of rules which allow a successful parse, find all the combinations of rules which produce successful parses.

Breadth-first / Depth-first / Backtracking or Lookahead – differing search strategies, pursue all options in parallel, follow one 'path' through the search space & backtrack if necessary, use (limited) lookahead into right context to make correct choice.

6.4 Shift-Reduce Parsing for CF PSG

Depth-first, no backtracking:
Stack (PDS) + Input Buffer (Queue)
Shift next lexical item in Input Buffer onto top of stack (with lexical syntactic category/ies)
Reduce one or more constituents in top two cells of Stack whenever they match a rule of the grammar (i.e. replace right hand side (RHS) of rule with LHS in stack)

```
Grammar                    Lexicon

S --> NP VP                 Sam, Hannah... : Nname
NP --> Nname                the, a.... : Det
NP --> Det N                boy, girl... : N
VP --> Vintrans             runs, smiles... : Vintrans
VP --> Vtrans NP            loves, hits... : Vtrans
VP --> VP PP                in, on, with... : P
PP --> P NP

Stack                                |  Input Buffer

---                                  | Sam hits the girl

(Sam, Nname)                         | hits the girl

(Sam, NP)                            | hits the girl

(Sam, NP) (hits, Vtrans)             | the girl

(Sam, NP) (hits, Vtrans) (the, Det) | girl

(Sam, NP) (hits, Vtrans) (the, Det) (girl, N)

(Sam, NP) (hits, Vtrans) (the girl, NP)

(Sam, NP) (hits the girl, VP)

(Sam hits the girl, S)
```

Is this algorithm bottom-up or top-down? How much use of the left/right context are we making? What are the conditions for success/failure with this parser? What would happen if we collapsed the distinction between Vtrans/Vintrans? What more would we need to do if we wanted to output the phrase marker? Is this algorithm guaranteed to terminate? What would happen if the grammar contained a rule such as NP $\rightarrow$ e, where e is the empty symbol (no input)?

Try some other examples:

The boy hit the girl with the shovel
The boy hit the girl with the shovel in the park

Can you design a similar top-down algorithm and parse the previous two egs? Do you notice anything about the rule VP $\rightarrow$ VP PP?

Shift / Reduce Parsing (breadth-first) – remove the Vtrans/Vintrans distinction from the grammar and define a similar breadth-first parser capable of correctly parsing our first e.g. What modifications do you need to make to the parser to make this possible? Now add the rule NP $\rightarrow$ NP PP and parse the same egs. What do you notice about the behaviour of the parser? How efficient is it?

6.5 CFG Worst-case Ambiguity

In the worst case, the number of analyses of a string of length n can be exponential 3^n (which is why any parser which enumerates all possible analyses will have exponential complexity).

A simple example is noun-noun compounding in English. The noun compounds in (18) all consist of a sequence of English nouns. The bracketings indicate which noun (compound) modifies which other noun (compound).

(18) a winter holiday
b winter holiday resort ((winter holiday) resort)
c toy coffee machine (toy (coffee machine))
d water meter attachment screw ((w m) (a s))
e airport long-term car park courtesy vehicle pick-up point
f (((airport (long-term (car park))) (courtesy vehicle)) (pick-up point))

Since only adjacent noun (compounds) can be in a modification relation the constraints on noun compounding can be expressed with one doubly recursive CF rule: N $\rightarrow$ N N – which licences all the binary branching trees with nodes labelled ‘N’ over the number of nouns in the compound. (Actually, strictly speaking we need to allow for some Nbar constituents, eg. *long-term*.)

The number of such binary braching trees is defined by the Catalan series, $C(n)$:

$$
\begin{aligned}
C(n) &= \left(\begin{array}{c} 2n \\ n \end{array} \right) - \left(\begin{array}{c} 2n \\ n-1 \end{array} \right) \\
&= \frac{2n!}{n!(2n-n)!} - \frac{2n!}{(n-1)!(2n-(n-1))!} \\
C(4) &= \frac{8!}{4!4!} - \frac{8!}{3!5!} \\
&= 70 - 56 = 14
\end{aligned}
$$

(where $n! = n(n-1)\ldots$ 1 ie. factorial)
The number of trees for a n noun compound is given by $C(n-1)$. Here is a table for n [3-8] with $(n-1)!$ shown for comparison:

n	3	4	5	6	7	8
$C(n-1)$	2	5	14	42	132	469
$(n-1)!$	2	6	24	120	720	5040

So (18e) has 469 analyses, but this represents a great deal of syntactic constraint over the 'free word order' case where any noun (compound) can modify any other non-contiguous noun (compound) – this is equivalent to $C(n-1)(n-1)!$ interpretations!

PP attachment is another example of worst case CFG ambiguity which follows the Catalan series – see Church and Patil, 1982 *Computational Linguistics*, 8, ' Coping with syntactic ambiguity or how to put the block in the box on the table' for more details.

6.6 Chart Parsing

Well-formed Substring Tables (WFSTs) – a way of avoiding duplicated effort in non-deterministic parsing. All efficient algorithms use WFSTs in some way. The basic insight is that there is no point in looking for and parsing the same phrase in the input repeatedly just because it features more than once in various different derivations (recall the breadth-first version of the shift / reduce parser).

The standard technique used in NLP to implement this idea is known as the 'chart'. So I will describe this. You might like to think though, how the same idea could be used with the 'stack' in a breadth-first shift / reduce parser.

A chart is a labelled, directed acyclic graph consisting of vertices and edges. The bottom of the graph contains the words of the input to be parsed. Vertices mark the boundaries of each word and of the complete input (the convention is 0 to n vertices). Edges span vertices and are labelled with syntactic categories. Edges which contain other edges are also labelled with the contained edges; i.e. VP[V,NP] labels a VP edge containing a V and NP edge, as illustrated informally in Figure 3 (where the containment information is not given for the NP edges).

This chart is incomplete. Can you add the S edge which represents the most plausible reading? The chart contains several edges which intuitively are not part of the final analysis (e.g. N, 1-2; NP, 2-7). In order to make this explicit and provide a way of recovering the phrase structure from the chart we need to annotate edges with the information about which other edges they contain (e.g. NP, 2-7 contains NP, 2-4 + PP 4-7). The chart inplements the WFST idea though, because it is possible for more than one edge to contain the same 'lower' edge (e.g. NP 2-4 is contained by NP 2-7 and by VP 1-4). The chart can represent global ambiguity too, since we could add further S and VP edges which contained NP 2-7 to get the semantically unlikely reading in which the PP is functioning adjectivally. Can you add these edges and then annotate all the edges in the chart to show which other

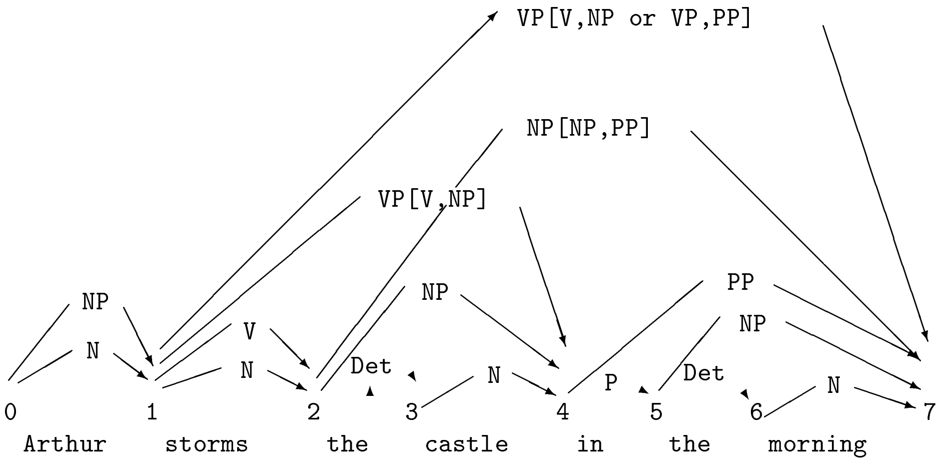


Figure 3: Chart Data Structure

edges they subsume using the grammar from the previous section?

The chart data structure is a method for sharing partial subanalyses between alternative partial or complete analyses. It is neutral between parsing algorithms (and grammar formalisms). Can you define a breadth-first algorithm which builds a chart using the grammar above? Can you show that it is more efficient than the shift / reduce, breadth-first stack-based parser you defined above?

Active Charts – WFSTs stop duplication of successful subparses but don't prevent the same hypothesis being tried many times over if it fails, because nothing gets put in the WFST. Active charts avoid this inefficiency by augmenting the chart to contain active (incomplete) and inactive (complete) edges.

Active edges encode hypotheses about the input to be tested. Inactive edges encode completed (sub)analyses. To add active edges, we augment the labels on edges from categories to 'dotted rules' and allow single cyclic arcs in the chart to represent active edges which (so far) haven't subsumed any other edges. The active charts in Figure 4 contain inactive lexical edges and add active edges representing the hypotheses about the input. As parsing proceeds some of these hypotheses are proved correct and the dot moves on through the RHS of the rules encoded in the active edges.

So far, we have considered the chart as a data structure, but now we need to explain how parsing proceeds in terms of active and inactive edges. **The fundamental rule** of chart parsing is: if the chart contains an active edge from i to j and an inactive edge from j to k (where $i \leq j < k$) then add a new edge from i to k if the category after the dot in the active edge matches the category of the inactive edge. The new edge should have the same label as the active edge with the dot 'advanced one' (where $A \rightarrow B\ C$. is equivalent to A).

Using this rule and the grammar above, see if you can simulate the behaviour of an active chart on a couple of the sentences we considered before. Assume that the fundamental rule is applied to every pair of adjacent active-inactive edge pairs and that active edges are added to the chart bottom-up keyed off the leftmost RHS category. Is this the most efficient way of applying the fundamental rule and adding active edges?

Different parsing algorithms / search strategies correspond to different strategies for adding active edges and different orders of application of the fundamental rule

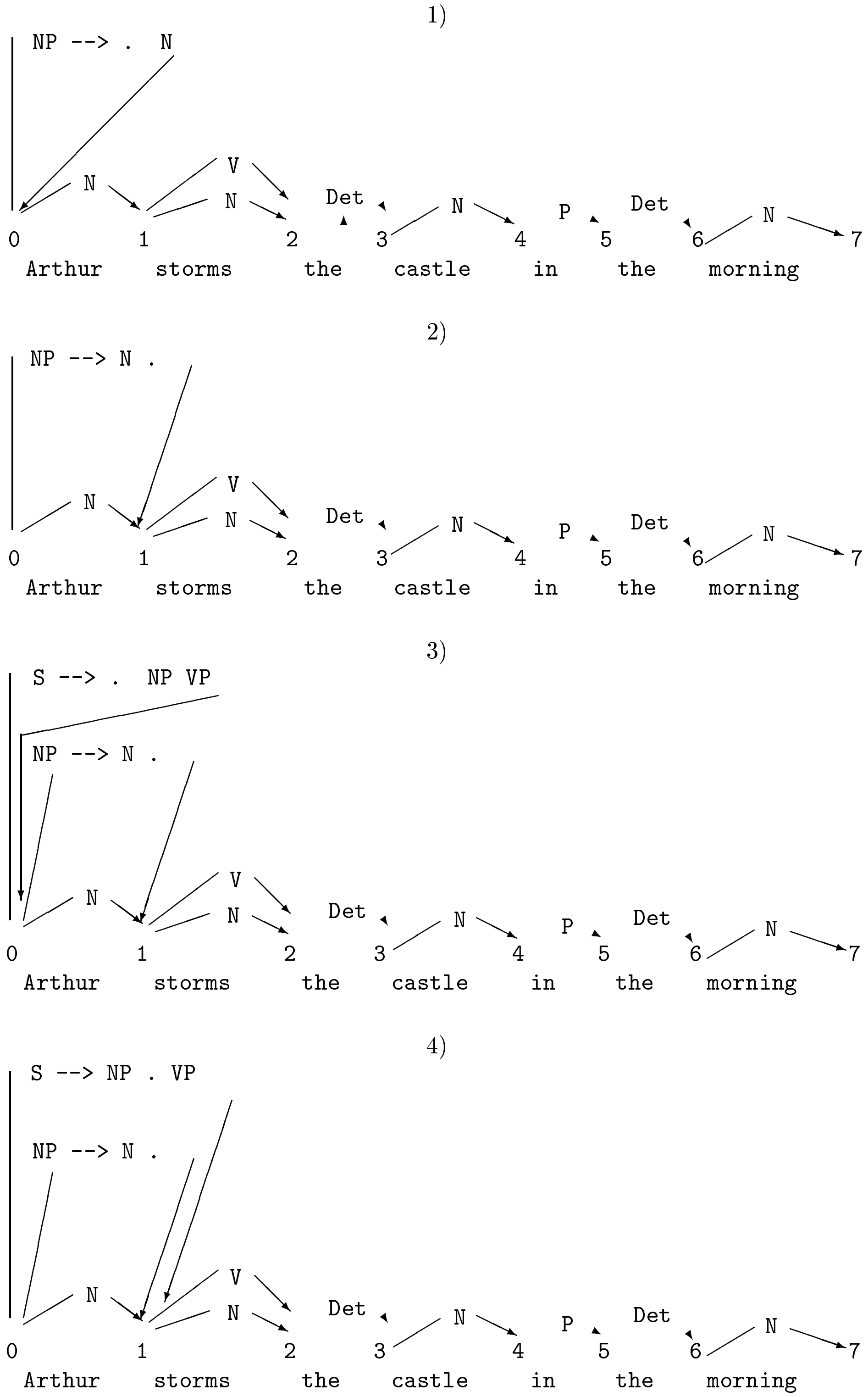


Figure 4: Active Chart Parsing

to active-inactive pairs. Chart parsers usually employ an **Agenda** which is a list of edges. When a new edge is created by an application of the fundamental rule, it is put on the agenda. When we get further edges to work on, if we treat the agenda like a stack (last in, first out) we get depth-first search, if we treat it like a queue (last in, last out) we get breadth-first search. Depending on how we add active edges to the chart, we can get top-down, bottom-up or a mixed parsing strategy. For instance if we index rules by their leftmost RHS category and add active edges to the chart 'bottom-up', whenever we build one of these leftmost daughters, and then parse depth-first from this new edge, we will get a 'left-corner' parser (which has some characteristics of both bottom-up and top-down parsers).

An edge is a record with a unique identifier (id), a 'to' and 'from' vertex, a category, a list of constituents sought, and a list found (i.e. the daughters in the corresponding (dotted) PS rule occurring before and after the dot). However, in order to recover the parse tree(s) these lists consist of the unique edge identifiers of the corresponding constituents / subanalyses:
edge(id, from, to, category, sought, found)

Left-corner Rule Invocation

if: edge(idN, fromX, toY, catA, _sought, _found) then:
for each rule: B $\rightarrow$ α
with leftmost category A in α
add to the chart an edge of the form:
edge(idN+1, fromX, toX, catB, sought-α, ())
and for each inactive edge:
edge(idM, fromX, to?, cat?, (), _found) add to the agenda:
<idN+1,idM>
(where N,M,X,Y are distinct variables, A and B are syntactic categories, and _... are anonymous or 'don't care' variables). In addition, each time an inactive edge is created by an application of the fundamental rule and added to the chart, <active,inactive> edge pairs must be added to the agenda. Can you work out the algorithm to do this? The only further requirements for a complete chart parsing algorithm are 1) a chart initialisation procedure which looks up the words to be parsed in a lexicon and creates vertices with inactive edges corresponding to the possible lexical syntactic categories for each word, and 2) a definition of success and a method for recovering each analysis from the chart. Can you see how to define these?

Chart Parsing Complexity – Sensible rule invocation strategies with a chart parser (and some other methods using WFSTs) gives (worst-case) polynomial complexity as a function of length n of input when using an ambiguous CF grammar (n^3). In parsing complexity results grammar size is treated as a constant factor and assumed to be non-significant. But if the grammar gets big enough it may be that the overhead of searching for appropriate rules etc. is so great that for 'normal' inputs (i.e. sentences of, say, up to 30 words) this is the real source of inefficiency.

Gazdar & Mellish ch6 describe chart parsing in more detail and discuss further issues such as recursion checking, proving search strategies complete, etc. Pulman describes a basic chart parsing algorithm in Prolog: 'Basic parsing techniques: an introductory survey.' *Encyclopedia of language and linguistics*, Asher, R.E. (ed) Pergamon Press, Oxford, 1993.

6.7 Parsing UB PSGs

Most systems make use of a context-free 'backbone' and add unification tests to an essentially context-free algorithm. Unification should add a constant factor to parsing complexity. Efficient implementations of unification involve destructive operations, so in a non-deterministic framework it is necessary to make sure you can recover the original categories in case they are needed again. The simplest way to do this is to copy each structure before applying the fundamental rule, but this is very space inefficient. There is a literature on more sophisticated schemes (see Gazdar & Mellish).

Things get more complex if the feature system is recursive – it means the grammar is certainly able to generate indexed languages which have parse complexity results of n^4 and it is quite easy to define parsing algorithms which will not terminate on some grammars (see Gazdar & Mellish again for more details).

See Carroll, 1994, 'Relating complexity to practical performance in parsing with wide-coverage unification grammars', *Proc. of 32nd Meeting of Assoc. for Comp. Linguistics* for a discussion of worst and average case parsing complexity for chart and related parsing techniques.

You can try out a 'head corner' chart parser using a grammar based on a formalism very similar to that introduced in section 5.11 at:
`http://odur.let.rug.nl/ vannoord/CL97/anltframes.html`

7 Semantics

We will consider word and sentence meaning from the perspective of logical or formal semantics since this provides an appropriate framework for doing automated inference. You may know the basics of proof theory and model theory for the propositional and first-order predicate calculus and the syntax of the (typed) lambda calculus – β-reduction. If you don't then it would be worth reading the relevant chapters in eg. Russell and Norvig, Ramsay A. *Formal Methods in AI*, CUP 1988. A good introductory book on formal semantics is Bach, E. *Informal Lectures on Formal Semantics*, State Univ of New York Press, 1989.

7.1 Semantic Intuitions/Evidence

Just as with syntax we use intuitions about 'grammaticality' to judge whether syntactic rules were correct, we will use our semantic intuitions to decide on the correctness of semantic rules. The closest parallel to ungrammaticality is nonsensicality or semantic anomaly. The propositions in (19) are all grammatical but nonsensical.

(19) a Colourless green ideas sleep furiously
b Kim frightened sincerity
c Thirteen is very crooked

Other propositions are contradictions, as in (20).

(20) a It is raining and it is not raining
b A bachelor is a married man
c Kim killed Mary but she walked away

The assertion of some propositions implies the truth of other propositions; for example (21a) implies b) and c) implies d).

(21) a John walked slowly
b John walked
c John sold Mary the book
d Mary bought the book from John

This relation is called *entailment* and is perhaps the most important of the semantic intuitions to capture in a semantic theory since it is the basis of the inferences we make in language comprehension and many other semantic notions reduce to entailment. For example, two propositions can be synonymous, as in (22), but the notion of synonymy reduces to the notion of identity of entailments.

(22) a John is a bachelor
b John is an unmarried man

We also have intuitions about the (semantic) ambiguity of certain sentences; that is they can convey more than one proposition, for example, those in (23).

(23) a Competent women and men go far
b He fed her dog biscuits
c Everyone knows one language

We would like our semantic theory to predict and explain these intuitions and thus we will use intuitions of this kind to evaluate semantic theories.

7.2 Truth-conditional Semantics

There are two aspects to semantics. The first is the inferences that language users make when they hear linguistic expressions. We are all aware that we do this and may feel that this is what understanding and meaning are. But there is also the question of how language relates to the world, because meaning is more than just a mental phenomenon – the inferences that we make and our understanding of language are (often) about the external world around us and not just about our inner states. We would like our semantic theory to explain both the 'internal' and 'external' nature of meaning.

Truth-conditional semantics attempts to do this by taking the external aspect of meaning as basic. According to this approach, a proposition is true or false depending on the state of affairs that obtain in the world and the meaning of a proposition is its truth conditions. For example, *John is clever* conveys a true proposition if and only if John is clever. Of course, we are not interested in verifying the truth or falsity of propositions – we would get into trouble with examples like *God exists* if we tried to equate meaning with verification. Rather knowing the meaning of a proposition is to know what the world would need to be like for the sentence to be true (not knowing what the world actually is like). The idea is that the inferences that we make or equivalently the entailments between propositions can be made to follow from such a theory.

Most formal approaches to the semantics of NL are truth-conditional and model-theoretic; that is, the meaning of a sentence is taken to be a proposition which will be true or false relative to some model of the world. The meanings of referring expressions are taken to be entities / individuals in the model and predicates are functions from entities to truth-values (ie. the meanings of propositions). These functions can also be characterised in an 'external' way in terms of sets in the model – this extended notion of reference is usually called denotation. However, we will do semantics in a proof-theoretic way by 'translating' sentences into FOL formulas and then passing these to a theorem prover since our goal is automated text understanding.

7.3 Syntax-driven Semantics

Another important aspect of meaning that we would like our semantic theory to explain is its productivity. We are able to interpret a potentially infinite number of sentences that convey different propositions. Therefore, just as in syntax, we will need to specify a finite set of rules which are able to (recursively) define/interpret an infinite set of propositions.

As the ambiguous examples above made clear, syntax affects interpretation because syntactic ambiguity leads to semantic ambiguity. For this reason semantic rules must be sensitive to syntactic structure. Most semantic theories pair syntactic and semantic rules so that the application of a syntactic rule automatically leads to the application of a semantic rule. So if two or more syntactic rules can be applied at some point, it follows that a sentence will be semantically ambiguous.

For example, if we assume a recursive rule like that below for coordination of sentences and lexical entries for *and* and *or* as shown, we can capture the ambiguity of examples like *Max snores and Belinda smiles or Felix purrs* depending on how we apply the rule.

$S \rightarrow S$ Conj S : Conj$'$(S'_1, S'_2).
and : Conj : and.
or : Conj : or.

The lexicon now provides a syntactic and semantic definition for each word and PS rules have a semantic component. The notation Conj$'$(S'_1, S'_2) is an instruction to apply the semantics of the second daughter on the RHS of the CF rule to the semantics of the first and third. The semantics of S will be a truth-value so we can think of the semantics of and/or as the standard truth-tables from propositional logic; that is functions from truth-values to truth-values. Can you show that you can get two different formulas for the example above with different semantics from the two possible phrase markers for it? Assume a standard NP VP structure and that (purr1 felix1), and so forth, are the formulas associated with each S. (Throughout, I will add numbers – usually 1 – to predicates and individual constants to remind you that these are not English words. The number on the predicate can be taken to indicate a particular sense of an English word form. That on the constants means a particular individual / entity.)

7.4 Compositionality

Pairing syntactic and semantic rules and guiding the application of semantic rules on the basis of the syntactic analysis of the sentence also 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. Can you demonstrate that we get an infinite sequence of interpreted sentences from a grammar incorporating the coordination rule above. 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 compositional in this way, then we cannot capture its semantic productivity (with a theory of this type).

7.5 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 (24).

(24) 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 (24) 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 (25) illustrates:

(25) 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)

Can you think of any three-place predicates in English? 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.

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. In fact, we will just 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.

What would the rule for a ditransitive verb (eg. *give*) look like?

7.6 NP Semantics

Not all NPs are names; (26) gives some examples of NPs with determiners and common nouns. Most determiners seem to function semantically like quantifiers in FOL.

(26) 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 (27a) and (26c), it should be clear that we can't capture their meaning by translating them into the FOL expressions in b) and d) respectively.

(27) 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? (27a) 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, (27c) 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 (28a,b), respectively.

(28) 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 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 in are logically equivalent, the third is the result of applying beta-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 beta-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)

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:

```
a)     S               b)                  t
     /   \                              /    \
   NP      VP                 <<e t> t>        <e t>
  /  \     |                  /       \          |
Det   N    V      <<e t> <<e t> t>>   <e t>    <e t>
 |    |    |          |                 |        |
Every man snores     all               man1    snore1
```

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.

7.7 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, (29a) only receives the interpretation (28b).

(29) 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 (29c). If you find this reading a little forced for (29a), there are other examples where it seems more natural, as (30) illustrates.

(30) 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, (31a) is ambiguous between b) and c), depending on the relative scope of *not* and *every*.

(31) 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). These examples 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. Does anything analogous occur in the semantics of logics or programming languages?

7.8 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 (32) to correctly characterise Kim's beliefs.

(32) 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 $\exists$ x miss-uk1(x) ..., so in the 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 Sandy 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) $\exists$ x Miss-uk1(x) $\wedge$ Believe1(Kim1 ^[$\neg$ Feminist(x)]
b) Believe1(Kim1 ^[$\exists$ x Miss-uk1(x) $\wedge$ $\neg$ 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 (33a) is true, then b) must be true (if we assume that *bachelor* just means 'unmarried man').

(33) 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. Can you see why? 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?

7.9 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 (34).

(34) 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 (35) it is implausible to argue that productive semantic rules combining the meanings of the individual words produce the most likely interpretation.

(35) 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. How might you deal with metaphorical phrases in the theory we've developed?

8 Statistical Parsing

The previous sections have outlined a knowledge-based approach to syntax, semantics and parsing. However, this approach tends to collapse in the face of the high degree of ambiguity and the amount of variation and change exhibited by natural languages. Many ordinary sentences of English contain multiple PPs, but these can be highly ambiguous for example, the sentence in (36).

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

has 5 different syntactic analyses and associated semantic interpretations given a standard grammar when there are 2 PPs. However, addition of a third PP increases this to 14 analyses. Even if we can compute these possibilities efficiently and represent them compactly using a technique like chart parsing, this does not bring us much closer to discovering the correct interpretation. The traditional knowledge-based approach to disambiguation was to try to capture and represent the knowledge about words in sentences like (36) which would allow the most plausible reading to be chosen; for example, that a telescope is an instrument for seeing with. However, this approach is very resource-intensive and has never been practical for anything other than very circumscribed domains.

A generative grammar consisting of a set of syntactic and semantic rules and associated lexicon characterises a language as a well-defined (infinite) set of sentences and defines a mapping from surface form to logical form(s) for each sentence of the language. This is an essentially static conception of a language which may be adequate, in principle, to describe the language of a single individual at a single point in their history. However, in reality a language is the joint output of many speakers with distinct linguistic backgrounds whose language varies from social context to context and across time in response to linguistic fashion and contact with speakers of different dialects. For example, English is now spoken by around 1.5K million people around the world either as a first language (e.g. in the USA) or as an official second language (e.g. in India) with considerable variation and degree of proficiency (e.g. size of vocabulary). On the web, examples of most dialects, genres and levels of such proficiency can now be found. Therefore, a NLP system incorporating a grammar developed manually by a small team of 'BBC English' speakers is unlikely to be able to return analyses for a high proportion of 'English sentences' found on the web. Furthermore, wide usage tends to speed up linguistic change, requiring that such a grammar be continually augmented. Manual development of state-of-the-art grammars of this type itself takes of the order of 10 person/years. Thus, such systems have mostly been deployed in circumscribed domains with a limited user base, such as database access or other system interface applications, and even here problems of coverage limit their viability.

Increasing availability of large quantities of textual or transcribed spoken corpora combined with increasing recognition of the inadequacies of the pure knowledge-based approach to NLP have led many researchers to explore more statistical, empirical and 'bottom-up' data-driven approaches.

8.1 Probabilistic Context-free Grammar

We can define a probabilistic version of a CFG (see section 5.8) in a manner analogous to the definition of a probabilistic FSA for PoS tagging (see section 4.4). A CFG consists of a finite set of productions or rewrite rules of the form $A \rightarrow \alpha$, where $A \in NT$ and $\alpha \in (NT \cup T)^*$. A PCFG is a CFG with a probability associated with each rule such that the sum of the probabilities of all distinct expansions of any given 'mother' category on the LHS of a rule, A, is one:

$\forall\ A \in\ NT, \sum P(A\ \rightarrow\ \alpha) = 1$

The probability of a given derivation is the product of the probability of the rules used in the derivation:

$P(D) = \prod_{R\ \in\ D} P(R)$

and the probability of a sentence is the sum of the probabilities of the derivations of that sentence:

$P(S) = \sum_{D\ \in\ S} P(D)$

Thus a PCFG defines a *language model* or stochastic language generator in which the probability of the language defined, the sum of the probabilities of all the sentences of the language or equivalently of all the derivations licensed by the grammar, is one (see e.g. Manning and Schutze, Charniak, or Allen for more detailed definitions, more formulas, some standard algorithms, and so forth). Typically, the parameters of a PCFG are estimated using maximum likelihood estimation from a training corpus of sentences annotated with correct analyses manually. Thus the probability of the training corpus is maximised:

$argmax_c P(C) = \sum_{S \in\ C} P(S)$

This reduces to estimating the probability of each rule of the grammar in terms of its relative frequency as an expansion of the mother category:

$P(A\ \rightarrow\ \alpha) = \frac{freq(A\ \rightarrow\ \alpha)}{freq(A)}$

We might use the probabilities of derivations to rank or choose between alternative derivations for the same sentence. For example, in (36) above the differing probabilities of the rules:

P1, NP $\rightarrow$ NP PP
P2, VP $\rightarrow$ VP PP

would determine which of the readings was most probable. However, this approach won't, in general, be very useful at picking out the derivations we want. Firstly, if P1>P2, then we will always prefer 'low' adjectival attachments of PPs to NPs regardless of the words in the sentence (and vice versa for P1<P2). Secondly, some distinct derivations will involve equal numbers of applications of these two rules but in different orders. (If you cannot see, this draw some derivations referring back to section 5 to find the other rules needed.) Given that the probability of a derivation is defined by the product of the rules used to obtain it, these distinct derivations and associated interpretations will have the same probability. Thirdly, unlike with the PoS tagging probabilistic FSA, the number of rules in CFG derivations for the same sentence can vary, for example, if some rules expand as two daughters and others as three for the same mother category. However, by taking the product we will always prefer the shortest derivation for a given sentence. Finally, we have seen that CFGs are not adequate as grammars for natural languages, so we would need to generalise the probabilistic model to feature-based unification grammar.

One way of resolving some of the problems identified above is to define a 'higher-order' PCFG in which, for example, the probability of a rule is conditioned on the 'ancestor' rule in the derivation which introduced the category to be expanded; for example, the probability of expanding a subject NP (i.e. the daughter NP in S $\rightarrow$ NP VP) as a pronoun is greater than the probability of expanding an object NP in this way. However, no structural technique like this can get away from the fundamental problem of PCFG, which is that, unlike the probabilistic FSA for PoS tagging, it does not take account of the words in the sentence. One way that one might try to take account of the words would be by replacing the terminal categories in CF rules with preterminals (i.e. words or lemmatised words); for example. the rule VP $\rightarrow$ V NP might be replaced by a set of rules VP $\rightarrow$ eat | love | ... NP, but this won't help much because if we are interested in modelling likely direct objects of these specific transitive verbs (e.g. *eat sweets*, *love Madonna* etc) then the probabilities of NP $\rightarrow$ sweets | Madonna | ... will be quite independent of the probabilities of the VP rules, given a standard PCFG. This won't be true of the higher-order model sketched above in which the NP rule probabilities will be conditioned on the VP rules. However, it is trivial to show that many lexical cooccurrences of this form extend across even larger stretches of a CF derivation; for example, in *children eat lots of sweets / broccoli* there are dependencies between *eat* and *sweets / broccoli* and between *children* and *eat sweets / broccoli*.

The process of assigning probabilities to a PCFG can be done via maximum likelihood estimation from an annotated corpus of sentences paired with correct derivations. This amounts to counting the number of times each mother category occurs in the corpus of derivations and the number of times it is expanded using a particular rule. The probability of a rule is the ratio of the frequency of the rule to the frequency of its mother category. PCFGs can also form the basis for grammar learning or induction. For example, suppose that we define a PCFG to consist of all possible rules of the form:

P, A $\rightarrow$ B C
P, A $\rightarrow$ a

where 'A', 'B' and 'C' are non-terminal categories and 'a' is a terminal category which cannot be expanded further. A PCFG of this form is said to be in Chomsky Normal Form and this restriction has two advantages; firstly, the set of such rules over a given finite category set is itself finite, and secondly, the length of alternative derivations for any given sentence is the same. A PCFG defined in this way with a uniform initial assignment of probabilities to rules defines the most 'liberal and uninformative' grammar over the category set and will not rule out any sequence of terminal categories. Suppose now that we have a PoS tagged corpus of sentences and that the PoS tagset is the same as the terminal category set of the PCFG. Then we can assign derivations to this corpus and count the number of times a particular rule is used in all derivations of all sentences in the corpus. These counts can be used to revise the initial probabilities of the rules. We can continue to iteratively revise rule probabilities by recomputing the revised probability of each derivation and then revising the probability of each rule again, according to the overall probability of the derivation in which it occurs. (The algorithm which is used to do this is an Estimation Maximisation algorithm called the Inside-Outside Algorithm. It is described in detail in Charniak and Manning and Schutze.) The effect of this process is to skew the probability of rules until they converge (i.e. stop changing up to some predefined threshold) at which point we have found a PCFG which defines a local maxima for the probability of the corpus. Now, any low probability rules which are not required to obtain the highest ranked derivation for any sentence could, in principle, be removed and the PCFG renormalised to define a smaller PCFG which models the corpus better.

Not surprisingly, this approach to PCFG induction does not result in grammars which assign correct derivations high probabilities. No information about correct derivations is available during parameter estimation so it is a matter of chance whether the resulting PCFG will favour derivations that group words in a linguistically-motivated way. Various attempts have been made to achieve useful PCFGs by assigning a higher prior probability to more linguistically-plausible rules or by providing more information during parameter estimation, such as the correct unlabelled bracketing of the sentences. However, the approach remains limited by the inherent problems of PCFGs discussed above.

8.2 Lexicalised Probabilistic Grammar

A fairly straightforward way of defining a lexicalised probabilistic grammar is to take a CFG and condition rule probabilities both on the mother category (PCFG) and on (some of) the words which are (indirectly) generated by that rule (PLCFG). For example, if we are generating (36) above, then the probability of various parts of one derivation might be represented by the following conditional probabilities:

P1(NP $\rightarrow$ Det N | NP, the, man)
P2(PP $\rightarrow$ P NP | PP, in, the, park)
P3(NP $\rightarrow$ NP PP | NP, the, man, in, the, park)

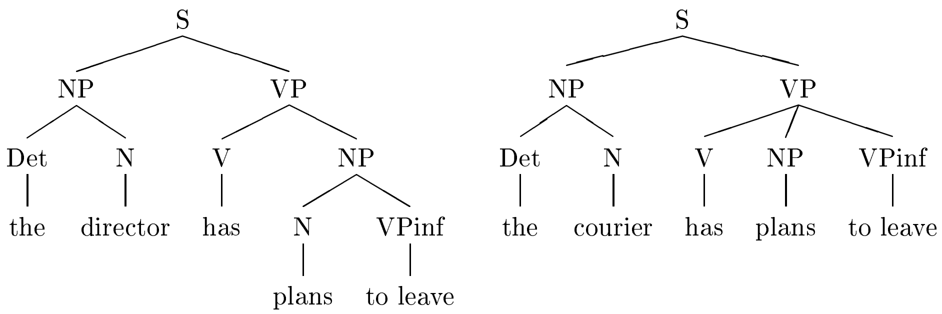
An approach like this certainly takes account of words, and now distinguishing adjectival from adverbial readings of PPs should be possible, in principle, because we will be comparing the probabilities of alternative rules conditioned on specific words, such as P4 and P5 below:

P4(VP $\rightarrow$ VP NP PP | VP, saw, the, man, in, the, park)
P5(VP $\rightarrow$ VP PP | VP, saw, the, man, in, the, park)

The problem is that any such model will now have a vast number of parameters. Whereas a large PCFG might have on the order of 10K parameters, corresponding to 10K rules. A PLCFG will have vastly more parameters bounded only by the size of the vocabulary of the language and the number of words that can be generated 'under' a single rule. What problem will this cause for parameter estimation?

There are several ways in which this problem might be ameliorated. We could reduce the number of words that we condition on by, for example, only conditioning on major categories (i.e. nouns, verbs, adjectives and prepositions) and perhaps by limiting the propagation of words up the derivation. For example, we could just condition on the head word of each daughter constituent in a rule, so S $\rightarrow$ NP VP would be conditioned on *Sam* and *saw* and not the whole sentence. We could also reduce word forms to lemmas/headwords (i.e. *saw* to *see*) so that the number of different word forms was reduced. Nevertheless, the resulting probabilistic model would still be vast.

A model of this form is still probably not optimal for selecting the correct logical form for a sentence. To do this, we need to model the probabilities of word *senses* entering into predicate argument relations. For an example like (37a), the most likely reading is b), but for c) it is d).



(37) a The courier has plans to leave
b ∃ (x,y) courier1(x) ∧ plans1(y) ∧ have1(x,y) ∧ leave1(x,y)
c The director has plans to leave
d ∃ (x,y) director1(x) ∧ have1(x,y) ∧ plans2(y,^leave2(x))

That is, the director makes a more plausible understood subject of the one-place leave2 predicate than a courier, who is more likely to be in a leave1 relation to some physical plans1. However, there is nothing in the two syntactic derivations corresponding to each of these readings which makes explicit the respective understood subjects of the two possible readings for *leave*:

8.3 The Problem of Sparse Data

Lexicalisation of probabilistic language models raises the issue of sparse data. Words have a highly-skewed, hyperbolic 'Zipfian' distribution. That is, the distribution approximates a power law of the form: $\frac{1}{RF^a}$ where $0 < a < 1$ and RF is the rank frequency of a word. So, for example, *the*, the most common word with a rank frequency of one will have a probability approximated by $\frac{1}{1^a}$, where a is around 1.3, and so forth. (Zipf showed that a 'doubly exponential' distribution of this form will approximate to a straight line when plotted on log-log scales with probability plotted against ranked frequency.)

This means that many words will have very low frequencies of occurrence, and if we sample text to train lexicalised models, there will always be a significant number of unseen words. Thus many of the parameters in a lexicalised model will receive a probability of zero if we use maximum likelihood estimation. Can you see why? The solution to this problem is to smooth the distribution obtained via maximum likelihood estimation with respect to some other 'background' distribution. One simple technique for doing this is so-called Add1 smoothing where we add one observation for each parameter in the model and then do maximum likelihood estimation. For example, in a PCFG this reduces to assigning each rule an initial probability of

$\frac{1}{|A|}$, for all $A \rightarrow \alpha$

This is equivalent to assuming a uniform prior distribution and then doing parameter estimation according to Bayes Law. The problem is that, as we have seen, the distribution of words is not uniform but highly-skewed, so this will not result in good estimates for unseen events (though it is better than assigning them a probability of zero).

More sophisticated smoothing techniques which do greater justice to the known underlying distribution require 'backing off' and conditioning on fewer variables in a systematic way. (Such techniques are described in Manning and Schutze, ch6 and Jelinek, F *Statistical Methods for Speech Recognition*, MIT, 1997, chap15.) However, in many situations it remains difficult to obtain good empirical estimates even for 'backed off' parameters. So the statistical approach might help improve the performance of NLP systems but it is not a panacea which leads directly to practical performance.

You can try out a much more robust parser which uses statistical techniques to pick the most likely analysis at:
`http://www.link.cs.cmu.edu/link/submit-sentence-regular.html`

9 Understanding (Discourse)

9.1 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 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). Furthermore, it is not possible to apply proof-theoretic rules in an unconstrained way to such a database. For example, consider what would happen if you freely applied the rule of $\wedge$-Introduction (p, q $\models$ p $\wedge$ q).

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 restrict itself to FOL analyses.

Our approach will look something like this: NL Semantics $\rightarrow$ [Beta-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 (38a) in our database and apply MPP and Universal Elimination, we will infer as many formulas like (38b) as there are men in the database and instantiate (38a) to as many useless formulas like c) as there are individuals (who are not men) in the database.

(38) a (all (x) (if (man1 x) (snore1 x)))
     b (snore1 Kim1)
     c (all (x) (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 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 (39a) and the database contains b) and c), then backward chaining will attempt to unify a) with all the atomic 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 atomic proposition with the identical formula c).

(39) 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 (40) means 'is there a value for _X which results in a true proposition?'

(40) (snore1 _Y)

In this case backward chaining will produce the sub-query (man1 _Y) which will unify with the atomic 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., some of which I hope you will (have) come across in other courses.

9.2 Speech Acts

The problem with our approach so far 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 simply assumes that all declarative sentences are assertions and all interrogative sentences are requests for information. Can you construct a situation / dialogue in which this would lead to a system misinterpreting some input?

An utterance conveys far more than a propositional content. Utterances are social acts by speakers intended to bring about some effect (on hearers).

Locutionary Act: the utterance of sentence (linguistic expression?) with determinate sense and reference (propositional content)
Illocutionary Act (Force): the making of an assertion, request, promise, etc., by virtue of the conventional force associated with it (how associated?)
Perlocutionary Act (Effect): the bringing about of effects on audiences by means of the locutionary act

What are the locutionary acts, illocutionary forces, and perlocutionary effects of the examples in (41)? Is this a sensible question?

(41) a Open the window
b The window is open
c Is the window open

These examples share the same propositional content but would have different illocutionary forces and perlocutionary effects in most contexts.

Natural languages do not 'wear their meaning on their sleeve'. Discourse processing is about recovering/conveying speaker intentions and the context-dependent aspects of propositional content. This relates back to the distinction we made at the beginning between decoding and inference. Decoding supplies a possibly underspecified logical form for an utterance which must be refined to a fully-specified formula representing the propositional content and illocutionary force of an utterance in context.

Speech acts have felicity conditions not truth-conditions and the former can't be reduced to the latter (Austin). Felicity conditions are constitutive for speech acts (ie. they are essential preconditions for an act to take place) (Searle). For example, you can't promise to do something unless you intend to do it, believe you can do it, wouldn't do it anyway, are being sincere, etc. Otherwise your act will be something other than a promising act.

So utterances have a 'force' as opposed to just a propositional content, and there are 'indirect' speech acts in which the force of an utterance is not that conventionally indicated by the grammatical mood (declarative, imperative, interrogative) of the sentence. Can you construct contexts in which the utterance of the examples in (42) would constitute an indirect speech act?

(42) a Would you pass the salt?
b Nuclear power is an ecological disaster.
c Shoot her!

Computation of the speech act intended by a speaker will be highly context-dependent, but essential to recovery of meaning in discourse, as the (very ordinary) exchange below illustrates:

A: That's the phone ringing. B: I'm in the bath.

Can you describe what is going on in terms of speech acts?

9.3 Deixis & Anaphora

We have been pretending that sentences contain enough information to allow some determinate proposition to be recovered from them, independently of context (modulo quantifier scope). However, the following examples demonstrate that many possible sentences of English do not (like this one!).

Deictic or indexical expressions are one traditional reason given for the need for a theory of pragmatics – a theory which by necessity must refer to language use and context. The examples in (43) exhibit person, place, and time deixis, respectively.

(43) a I am hungry
b Will you shut the window
c That's the shop
d You catch the bus over there
e I didn't have a PhD then.
f I'll see you on Wednesday

In each case the propositional content is unclear until it is fixed by the extra-linguistic context of utterance. Most deixis is reducible to truth-conditional meaning, so Montague, Lewis, Stalnaker, etc. have proposed reformulations of possible worlds semantics which treat propositions as functions from possible worlds *and contexts* to truth-values, or alternatively sentences as functions from contexts to propositions and propositions as functions from possible worlds to truth-values. Context is treated as a set of indices, coordinates or reference points, for speakers, addressees, times, places, and so forth. (In fact, some types of deixis, eg. social deixis, *Sir*, etc., are probably not reducible to truth-conditions.)

Anaphora occurs when a linguistic expression (usually a pronoun) is coreferential with a previous expression and where the semantic content of the anaphor (pronoun, definite NP, etc) is sufficiently vague that it can only select a referent by virtue of the preceding linguistic context (ie. the extra information specified by the antecedent), as in (44).

(44) a Kim thinks that he is clever (he=Kim)
b Sandy likes cars. She bought a Maserati last week (She=Sandy)
c Volvo drivers who wear hats believe that they own the road (they=Vds+hts)
d The house was empty. The front door was broken. (front door = front door of house)

The class of linguistic expressions which can function deictically or anaphorically overlaps substantially (creating ambiguity). Definite NPs, as well as pronouns, often function anaphorically linking back to previously introduced discourse referents which are either less accessible (e.g. 'further back') in the discourse or require some additional inference to make the link. For example, in (44d) *The front door* is coreferentially linked to *The house* via a so-called 'bridging' inference that houses (mostly) have front doors. Less frequently, anaphors precede there 'antecedents' usually in 'marked' circumstances, as in (45). (The traditional (but largely unused) term for this is cataphora.)

(45) a He was tough. He was good. He was handsome. Superman was going to save Hollywood.
b After she had thought it through, Sandy decided to do linguistics.

Determining antecedents for anaphors appears to require general knowledge, as (46) shows.

(46) a The men allowed the women to found their club. (their = women)
b The men allowed the women to join their club. (their = men)

9.4 Discourse Structure

Discourse has an information structure, discourses are about a topic, many phenomena like anaphora are resolved via this information structure. For example, below the discourse topic is initially Kim and Sandy and then switches to their transport arrangements.

a) Kim and Sandy are schoolteachers. b) They live in Carshalton Beeches and work at the same school. c) She drives him there every day, but d) he is taken home by a colleague when Sandy has to take games. e) On winter mornings, Sandy's car often will not start. f) She owns an old Volvo estate, but g) she frequently borrows her mother-in-law's Metro. h) It was her mother-in-law who sold her the Volvo, i) so she feels guilty when it doesn't work.

The unmarked organisation of a discourse is as a set of sentences with given information preceding new information – b), c), d) above. Given information is naturally pronominalized, ellipsed, etc.

Theme/Rheme or Topic/Comment are terms often used to talk about this level of linguistic organisation. These terms are distinct from Subject/Predicate and Given/New. For example, theme is defined as "the communicative point of departure for the rest of the clause". In b) *They* is grammatical subject, theme, and given information. In e) *On winter mornings* is new information, not grammatical subject, and therefore a 'marked' theme (Quirk etal, *Comprehensive Grammar of the English Language* Longman, 1985:ch18). Passives can function to 'thematise' a NP which cannot occur as subject otherwise; eg. in d) *he* is patient ('takee' not 'taker') of *take* and would therefore normally be the object. This is natural here because *he* is given and *a colleague* is new information.

Focus is a term used to refer to the linguistic expression which conveys the information which is the focus of attention. This can be signalled by (nuclear) stress, or by particular syntactic constructions; for example, in h) focus is on *her mother-in-law*. This is a marked situation, in normal cases focus is often on all the VP/new information – wide vs. narrow focus. Focus extends 'backwards' from nuclear stress up to but not including the theme. Nuclear (roughly strongest) stress usually occurs on the last contentful (stressable) word of the sentence.

Focus has been used as the basis for an algorithm to find the antecedents of anaphors by Sidner (see Brady, M. & Berwick, R. 1983 *Computational Models of Discourse*, MIT, for more detail). Sidner's theory of local focus tracking and maintenance is based on the close ties between anaphora and focus – whatever is in focus is referred to anaphorically, anaphors which don't refer to current focus change the current focus

1) Determining Expected (Initial) Focus
a) Subject NP if sentence is a copular (*be*) construction (eg. *JOHN is a doctor*)
b) Object of sentence (affected object) or object of sentential complement (eg. *John hit MARY*)
c) Other verb argument positions in order of preference with subject last (eg. *John gave a present TO MARY*)
d) The VP itself

Expected focus is (dis)confirmed after first anaphor. Whatever is antecedent of that anaphor remains current focus and 'first choice' as antecedent. Selection of potential antecedent is done via focus, evaluation is done via agreement & inference – focus is a technique for guiding inferential processing.

Sidner's algorithm works by maintaining a current focus (CF), an actor focus (AF), an alternative focus (ALF), and a stack of old foci. Initially, the CF is set to the expected focus and the ALF to another potential focus. The CF changes when inference rejects the CF as a possible antecedent. In that case the CF is set to the ALF/AF or a new element which is the antecedent and the CF is pushed onto the stack. The actor focus is set to the syntactic subject/agent (not copular constructions).

	AF	CF	ALF	Stack
1 Last night we went to a pub	we	pub		
2 It was pretty crowded	/	pub		
3 The beer was awful	/	beer		pub
4 We drank lager	/	beer	lager	pub
5 I prefer it to beer anyway	I	lager	/	beer, pub
6 Susie drinks it by the gallon	Susie	lager	/	beer, pub
7 Her liver must be a mess	/	Susie	liver	lager, beer, pub
8 When she dies,	/	Susie	liver	lager, beer, pub
9 they'll have to beat it to death	they	liver	/	Susie, l, b, p.
10 Anyway by the time we left the pub	we	pub	/	liver, S, l, b.
11 it was almost empty	/	pub	/	liver, S, l, b.

Initially, focus is on the pub and the first pronoun refers to it. Sidner's algorithm does not capture the 'bridging' inference from pub to beer which licenses the use of a definite NP in the third sentence. Lager becomes the alternative focus at 4 because no pronoun forces change of current focus, but in 5 *it* must refer to *lager* so this becomes current focus. The narrative continues with switches between AF, ALF and CF until 10 when there is a bigger 'shift' to the initial focus, the pub. I've assumed that the parsing system is able to recognise the non-referential nature of *gallon* and *mess*. Most changes of focus are predicted to be between AF or ALF and CF but they can involve popping old foci off the stack in a 'last in, first out' fashion – this predicts a 'hierarchical' discourse structure. The eg. above violates this, but this may be why a definite NP anaphor is needed – *Anyway... the pub*. Many dialogues appear to have a hierarchical structure which mirrors, say, a hierarchically decomposable task, such as assembling a water pump. The weakness of the approach lies in the arbitrary amount of fine-grained inference potentially required to recognise a shift of focus / coreference.

9.5 Intentionality

Not all discourses exhibit the type of discourse structure exemplified above. For example, below is a perfectly coherent discourse which contains no explicit anaphoric links. Its coherence derives from recognising the intentions of the participants:

A: Pint, please. B: Bitter? A: Tetleys. B: 1 pound 80 please. A: Thanks.

Therefore, other researchers have argued that structure is a side-effect rather than essential clue to discourse coherence and have explored the possible discourse 'moves' which can be made – rhetorical/discourse coherence relations. For example, (47b) is intended as an elaboration of a) and we resolve the links between the two sentences because we recognise it as such (not because of structural clues such as focus).

(47) a Kim can open the safe.
b He knows the combination.

Other relations include *narrative*, the default, *explanation*, *contrast* etc (see discussion in Allen, and further references).

9.6 Defeasibility

On his way to school, Kim worried about yesterday's Maths class. He felt it had gone badly.

Is Kim a teacher or pupil? Most assume teacher, at this point. However, either continuation in (48) is possible.

(48) a He decided he would try and explain sets again.
b He decided he would apologise for annoying the teacher.

These types of inference are not deductive because they are defeasible – plausible inference, abductive inference, non-demonstrative inference, non-monotonic logics. (Different from induction – see discussion in Russell and Norvig.)

9.7 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 (49).

(49) 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 (see eg. Charniak & McDermott, *Intro to AI* 1985, ch8, or Russell & Norvig) 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 (50a) can be interpreted using abductive inferences.

(50) 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 logic-like representation (in which variables are implicitly universally quantified) of how to do this (following Hobbs). 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. Can you see any disadvantages to this technique as a general approach to discourse understanding?

9.8 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.

Planning is a well-studied and independent area of AI used in robotics, decision support, etc. Given a Goal and an Initial state, how to get from I to G? Decomposition into elementary steps, preconditions of each step, dependencies between each step, finally sequencing each step (see Charniak & McDermott, 1985:ch9 for more details on planners).

Plans are made up from action schemas. An action schema consists of a set of preconditions which must be true if the action is to succeed, a set of effects which will become true after the action is executed, and a body/decomposition which specifies the action in more detail (ie. a sequence of actions or further intermediate goals to be achieved). Action schemas have parameters, such as Actor, which are instantiated to form particular action instances. Plans consist of a sequence of action instances.

Plans are often constructed using backward chaining and world states (goals, initial states) are expressed as sets of propositions. Preconditions, effects, etc. are also stated as propositions. Our interest in plans and planning is in the use that we can make of them to simulate the production and comprehension of purposeful discourse. Production = plan production, Comprehension = plan recognition. Selecting a plan and interpreting text relative to it is an abductive inference about the speaker's intentions and goals.

The diagrams show an example of some action schemas relevant to discourses about train journeys taken from Allen (1987:379f), and some of the inferences that can be made on the basis of 'decomposition' chaining through such plans on the basis of matching sentences to parts of plans. For example, given the plan knowledge base below if a story starts with (51a), and the parser produces logical formulas like b), then this can be used to instantiate the purchase action schema.

```
TAKE-TRIP(Actor, Train, Destination)
preconditions: DESTINATION(Train, Destination)
add (result): IN(Actor, Destination)
decomposition: PURCHASE(Actor, CLERK, Ticket(Train)) ->
               GOTO(Actor, Loc(Train)) ->
               GET-ON(Actor, Train))

GOTO(Actor, Loc)
preconditions: NOT(AT(Actor, Loc))
add (result): AT(Actor, Loc)

GET-ON(Actor, Train)
preconditions: AT(Actor, Loc(Train)), HAS(Actor, Ticket(Train))
add (result): ON-BOARD(Actor, Train)
decomposition: GIVE(Actor, Conductor(Train), Ticket(Train)) ->
               FIND-SEAT(Actor, (any SEAT))

GIVE(Actor, Recipient, Object)
precondition: HAS(Actor, Object)
delete (result): HAS(Actor, Object)
add (result): HAS(Recipient, Object)

PURCHASE (Buyer, Seller, Object)
preconditions: HAS(Buyer, Price(Object)), HAS(Seller, Object)
```

```
delete (result): HAS(Buyer, Price(Object)), HAS(Seller, Object)
add (result): HAS(Seller, Price(Object)), HAS(Buyer, Object)
decomposition: GOTO(Buyer, Loc(Seller)) ->
               GIVE(Buyer, Seller, Price(Object)) ->
               GIVE(Seller, Buyer, Object)
```

(51) a Jack went to the ticket clerk.
b GOTO (JACK1, Loc(CLERK))
c P1: PURCHASE(JACK1, Ticket(Train), CLERK)
d T2: TAKE-TRIP(JACK1, Train(T2), Destination(T2)

(Note that it won't instantiate the take-trip schema because Allen's logical formulas have semantic restrictions on predicate arguments.) The action instance P1 with parameters as in c) will in turn cause the creation of a take-trip action instance d). (Notice that Allen is using functions, such as Actor, which return individual constants, so Actor(P1) = JACK1 etc.)

The partially instantiated plans now create expectations and further sentences can be interpreted relative to these expectations using the decomposition chaining technique. Can you sketch how might this work for the discourse below?

Jack wanted to take a train to Rochester. He gave the clerk the money.

Can you see any problems with the general approach?

(See `http://www.cs.rochester.edu/research/trips/movies/` for a demonstration of this approach to dialogue understanding.)

9.9 Statistical Discourse Processing

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, prototype 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 robust statistical parsing of the kind described in section 6, the third by simply indexing the linear position of each identified NP, and the fourth is approximated by making use of superficial textual clues such as paragraph boundaries and cue words for discourse segment boundaries (e.g. *anyway* in the 'pub discourse' of section 8.4).

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. Notice that the salience factors are not so different from the focus rules employed by Sidner's algorithm in section 8.4, but the inference requirement to reject coreference has been replaced by numerical weights set experimentally. (For further details see Kennedy & Boguraev in Proc. of Coling96 and ECAI96 and Lappin & Leass, 'An algorithm for pronominal anaphora resolution' *Computational Linguistics*, 20.4, 535–61, 1994.)

Once a set of coreference classes has been discovered, it is fairly easy to see how a document can be transformed into a structure which represents what has been predicated of any given discourse referent, This provides a much more satisfactory starting point for extracting documents relevant to user queries, or for summarisation of their content. Much of current NLP research is focussed on sidestepping the 'knowledge bottleneck' which results from the (purely) knowledge-based perspective on NLP by using 'shallower' statistical approximations determined from empirical work with large corpora. Recent work on anaphora resolution of the kind outlined in this section has provided a striking demonstration of the possible benefits of a more empirical, statistical approach to NLP (sub)tasks.

The End

Lectures Appraisal Form

If lecturing standards are to be maintained where they are high, and improved where they are not, it is important for the lecturers to receive feedback about their lectures. Consequently, we would be grateful if you would complete this questionnaire, and either return it to the lecturer in question, or to the Student Administrator, A201 Computer Laboratory Tower, 2nd floor. Thank you.

1. Name of Lecturer: `Dr Ted Briscoe`

2. Title of Course: `Natural Language Processing`

3. How many lectures have you attended in this series so far? ....

   Do you intend to go to the rest of them? Yes/No/Series finished

4. What do you expect to gain from these lectures? (Underline as appropriate)

   Detailed coverage of selected topics *or* Advanced material
   Broad coverage of an area *or* Elementary material
   Other (please specify)

5. Did you find the content: (place a vertical mark across the line)

   Too basic ——————————————— Too complex
   Too general ——————————————— Too specific
   Well organised ——————————————— Poorly organised
   Easy to follow ——————————————— Hard to follow

6. Did you find the lecturer's delivery: (place a vertical mark across the line)

   Too slow ——————————————— Too fast
   Too general ——————————————— Too specific
   Too quiet ——————————————— Too loud
   Halting ——————————————— Smooth
   Monotonous ——————————————— Lively

   Other comments on the delivery:

7. Was a satisfactory reading list provided? Yes/No
   How might it be improved.

8. Apart from the recommendations suggested by your answers above, how else might these lectures be improved? Do any specific lectures in this series require attention? (Continue overleaf if necessary)