CPS331 Lecture: Natural Language

last revised March 12, 2012

Objectives:

1. To understand the phases of analysis of natural language input to a program
2. To understand how a grammar can be represented using Prolog DCG notation

Materials:

1. Projectable of an dictionary entry showing phonemes
2. Projectable of alternative pronunciations for tomato
3. Projectable of DCG examples
4. ParseTreeDisplayer program and handout of grammar
5. Demo of natural language blocks world program
6. Demo of natural language isa program
7. Discussion guide for Searle article

I. Introduction

A. Working with natural language has long been an area of interest in AI.

1. By "natural language", we mean human languages (such as English), as opposed to artificial languages such as programming languages.

2. Recall that Turing's article that we read at the start of the course made the ability to converse in natural language the test for machine intelligence.

3. Much of the work in knowledge representation (e;g. the development of semantic nets) was done in conjunction with work in this area.

4. Natural language understanding has been called an "AI complete" problem - meaning that truly solving this problem is equivalent to achieving strong artificial intelligence. (Where "truly solving" may mean much more than what Turing proposed ...)

No one would claim that any existing system comes close to such an achievement, of course.

5. On the other hand, there are many practical systems which use a solution to a small subset of the overall problem to accomplish limited ends.

   Examples?

   ASK

   a) Telephone speech recognition

   b) Queries to search engines

   c) Google language tools

B. The "Natural Language problem" actually encompasses many issues and subproblems.

1. Practical systems versus exploring general theories.

2. Natural language "understanding" versus natural language generation

   As we shall see, using the term "understand" to describe an AI natural language system is quite controversial. Perhaps "analysis" is a less controversial term, and one we will sometimes use.

3. Spoken versus written language.

   a) A subcategory of the "understanding" problem is speech recognition

Though it is possible, in principle, to build a system that uses speech recognition as a "front end" for a more complete understanding system, most practical applications of speech recognition have much more restricted aims.

Example: a telephone banking system can handle a set of standard banking transactions, but other queries must be handled by a human.

b) When we start with written text, more sophisticated aims can be achieved.

Example: Google's translation facilities.

4. Your book focuses on understanding of written text after a brief discussion of speech recognition, as shall we, though we will also say a bit about generation.

Actually, there is another problem the book does not discuss: speech synthesis - which is to natural language generation what speech recognition is to natural language understanding We will not discuss this, though.

C. The problem of understanding written text can, in turn, be divided into several subproblems.

1. Syntax or parsing

2. Semantics

3. Pragmatics

4. Examples

a) Consider the following English sentence "Can you tell me the time?

(1) Syntactically, the subject is “you”, the verb is a verb phrase composed of the auxiliary verb “can” plus the main verb “tell’, the indirect object is “me” and the direct object is “the time”

(2) Semantically, it is a question about the hearer/reader’s ability to report the time of day

(3) Pragmatically, it is actually a request for the hearer/reader to do something - i.e. the expected answer is not “yes” but something like “it is < fill in current time >”

b) Now consider the following English sentence: “colorless green ideas sleep furiously”

(1) Syntactically, the subject is ideas, modified by the adjective phrase “colorless green”, and the verb is “dream” modified by the adverb phrase “furiously”.

(2) Semantically, it’s meaningless! Hence, a natural language understanding program would probably not consider pragmatics at all.

c) Now consider the following English sentence: “Time the 9 is AM”.

(1) Syntactically, it is garbled. Hence, a natural language understanding program would probably not consider semantics or pragmatics at all.

(2) Of course, it might still be possible for a human to extract some meaning from it - especially if it were given in answer to our first question.

(3) That is, though it is syntactically unrecognizable, for a human it may well have both semantic content and pragmatic usefulness.

5. The separation into subproblems is not rigid, though.

   a) Consider the following phrase: “the highest student’s grade”

      (1) Syntactically, “highest” could modify either “student’s” or “grade”.

      (2) Semantically, the latter interpretation of the syntax would mean the phrase is talking about the best grade from a list of grades, while the former interpretation of the syntax would mean that the phrase is talking about the grade earned by the biggest drug user!

      (3) Pragmatics might serve to guide the syntactic analysis - does the phrase occur in a context of discussing academic achievement or discussing student drug use?

   b) If speech recognition is included, then the interrelationship between the subproblems becomes even stronger. For example, the English words “red” and “read” are pronounced exactly the same way. (In a dictionary, the pronunciation is denoted as “red”).

      The correct interpretation of “red” in “John read the book” requires analysis of Syntax (read can be a verb but red cannot).

      But in the sentence “John is well red” the correct interpretation requires analysis of Semantics as well

II. **Speech Recognition**

A. The general problem of speech recognition is way beyond the present state of the art. Practical systems succeed by limiting the problem in some way(s).

1. Limited vocabulary versus large vocabulary systems

   A limited vocabulary system expects input to be chosen from a small number of words.

   Example: many telephone systems are designed to handle the 10 digits plus a few words like “yes” or “no” (or synonyms like “OK”) Some will also recognize words like “human” or “operator” as a request to speak to a human - though I’ve found that sometimes when I want to speak to a human just saying over and over again “I want to speak to a human” to a system that does not recognize the phrase causes the system to give up on trying to understand me so it gives me what I want!

2. Single-speaker versus speaker-independent systems

   a) A single-speaker system is intended to be used by a single user, and is “trained” by the user to recognize his/her voice.

      Example: dictation software

      Large vocabulary systems are typically single-speaker

   b) A speaker-independent system is intended to be used by any user, with no training required.

      Example: telephone banking

      Speaker-independent systems are typically limited vocabulary

3. Continuous-speech versus individual-word

   a) In ordinary speech, we tend to slur words together, and the pronunciation of a word often depends on what occurs before and after it. Thus recognizing continuous speech is a particular challenge.

      Example: the following two sentences sound very similar, but are actually quite different:

      "How to recognize speech"
      "How to wreck a nice beach"

   b) Many speech recognition systems (particularly speaker-independent ones) therefore require the user to say each word individually. (This is not unlike trying to communicate with a native speaker of another language who is learning English, BTW - e.g. my China experience)

B. Speech recognition systems must address two issues: recognition of phonemes, and recognition of words.

1. A phoneme is a single basic sound.

   a) A dictionary may indicate the pronunciation of the word by giving the appropriate series of phonemes.

      Example: PROJECT Dictionary entry showing phonemes

   b) All human languages use a limited set of 40-50 phonemes - but the set is different for different languages.

      This often constitutes a challenge when learning a different language; it may be hard to say certain of the phonemes that don't occur in one's native language, or even to hear the difference between two phonemes that don't occur in one's native language - e.g. the difficulty Japanese speakers often have with "l" and "r" in English.

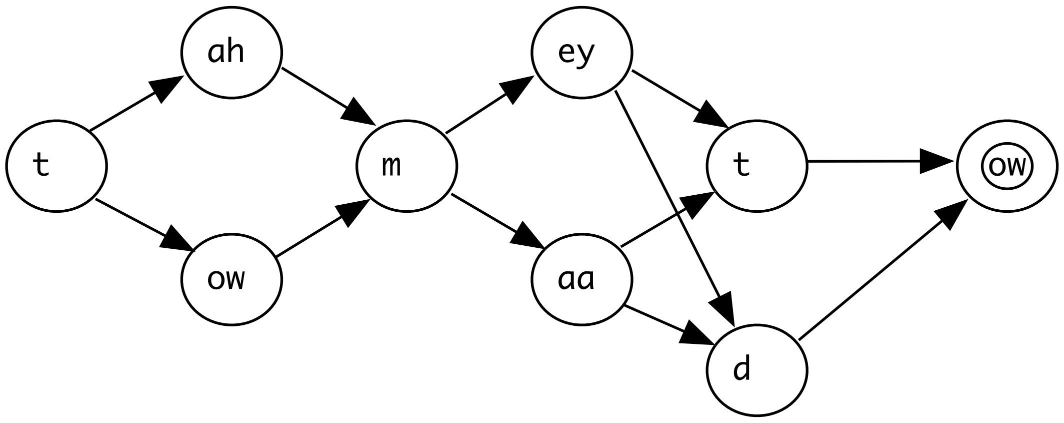
c) Different speakers may pronounce a given phoneme slightly differently. and may say two different phonemes in a way that is hard to distinguish.

2. A word, in turn, is composed of a sequence of phonemes.

   a) But different speakers may use different phonemes when saying the same word.

   Example: there are lots of different ways of saying "tomato"

   PROJECT



(Adapted from Luger, *Artificial Intelligence* 6th ed p. 186 - some technical phonemic symbols simplified)

PROJECT

   b) In continuous speech, it may not be clear where the phonemes of one word end and another begins

c) Moreover, in continuous speech slurring of words may result in a phoneme for a word being omitted

C. Most successful large vocabulary systems rely heavily on statistical techniques using Hidden Markov Models.

These use information about the probability of occurrence of various sequences of phonemes and words to help identify the most probable interpretation of an input sequence.

III. **Syntax**

A. We have already noted that analysis of syntax cannot be divorced from the rest of the steps in natural language understanding.

1. Recall the earlier example of "the highest student's grade"

2. Here's another classic example - consider the two English sentences:

   "Time flies like an arrow"

   "Fruit flies like a banana"

   Though the middle three words are the same in both cases, syntactic analysis will produce quite different results

   a) In the first sentence "flies" is the verb and "like" is a preposition

   b) In the second sentence, knowledge of the behavior of fruit suggests that we interpret "flies" as a noun modified by "fruit" and "like" as the verb. (Though the first analysis might be possible if one were talking about a food-fight :-))

3. Nonetheless, for simplicity we are going to treat syntactic analysis separately.

B. Syntactic analysis typically makes use of a formal grammar for the language.

1. A formal grammar spells out the rules for constructing "sentences" in a given language (where we use "sentence" generically to refer to whatever kind of linguistic entity is being specified by the grammar.)

   We way that a sentence belongs to a language if and only if it conforms to the grammar for that language

2. A formal grammar can be used in two ways:

   a) Synthetically - to construct sentences.

   b) Analytically - to analyze a sentence to see if belongs to the language defined by the grammar and - if so - how it was constructed according to the rules of the grammar.

3. Formal grammars are constructed from two kinds of symbols

   a) A terminal symbol is one that might appear in a sentence constructed according to the grammar.

      Example: English - each of the words appearing in the dictionary would be considered a terminal symbol, as would the various punctuation marks we use.

   b) A non-terminal symbol is one that wouldn't typically appear in an actual sentence constructed according to the grammar, but instead is used as part of the process of defining the grammar.

Example: English - sentence, clause, phrase ...

(Of course, a non-terminal might appear in a sentence talking about the grammar - but in that case it's being used as a terminal symbol)

c) One of the non-terminals is designated as the "start" symbol - which means that's what you start with in applying the productions.

Example: English - often "sentence"

4. As was the case in Cawsey, we are going to make use of a feature of the Prolog programming language known as Definite Clause Grammar (DCG), though more sophisticated strategies would be used in sophisticated systems. (DCG's have the advantage of being very easy to understand.)

A DCG is a collection of grammar rules or productions - each of which describes a way of producing a particular grammatical construct.

Example:

```
sentence --> noun_phrase, verb_phrase, noun_phrase.
```

is the DCG way of saying "one can produce a sentence by producing a noun phrase and then producing a verb phrase and then producing a noun phrase.

a) The left-hand side of a grammar rule always consists of a single non-terminal symbol. Non-terminal symbols name grammatical categories - e.g. sentence, noun_phrase, verb_phrase.

b) The right-hand side is a sequence of symbols, which can be any combination of non-terminal and terminal symbols.

Example: `noun_phrase --> [ the ], noun.`

says that a noun phrase can be produced by using the terminal symbol the, followed by a noun.

(1) A terminal symbol is represented by a Prolog list - e.g. [the].

(2) Actually, it is also possible to have a list of terminals - e.g. [will, give] matches the two-word sequence “will give”.

c) A grammar rule always ends with a period.

d) Any non-terminal symbol that appears on the right-hand side of a grammar rule must be defined by one or more productions. It is permissible for there to be several productions for the same non-terminal symbol

Example:
```
noun_phrase --> determiner, noun.
noun_phrase --> proper_noun.
```

specifies two different ways of producing a noun phrase.

In a complete grammar, there would also need to be productions for verb_phrase, noun and proper_noun, of course.

e) The choice of names for non-terminal symbols is totally arbitrary. The choices in the example were based on using names that would be meaningful to a human reader of the grammar, while minimizing typing. But exactly the same behavior would result from the following grammar:

```
sentence --> x, y, x.
x --> [ the ], z.
...
```

(where now we would need a grammar rule for “z” instead of “noun”). Formal grammars are frequently specified using production rules. There are different categories of grammars (of varying levels of power) that correspond to different sorts of requirements imposed on the production rules.

5. The process of analyzing a sentence using the rules of a grammar is called parsing. We can represent the fruit of such analysis using a structure called a parse tree.

   a) Suppose we have the following very simple grammar for a small subset of English:

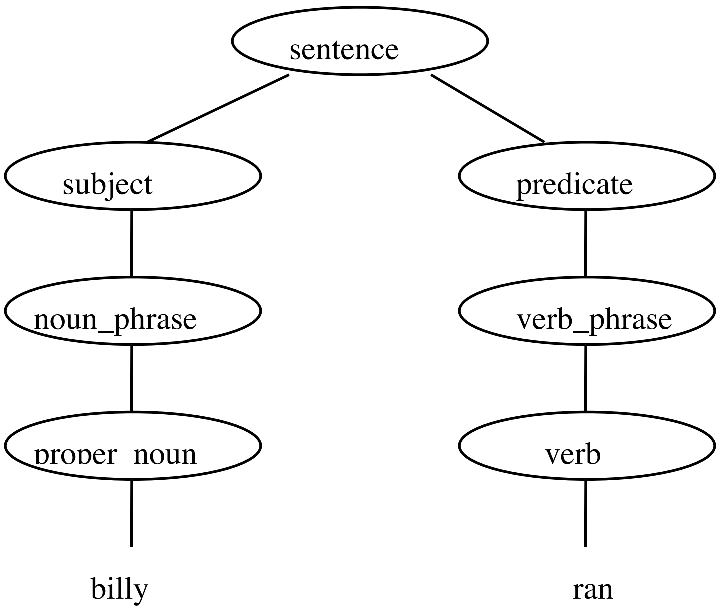
      PROJECT

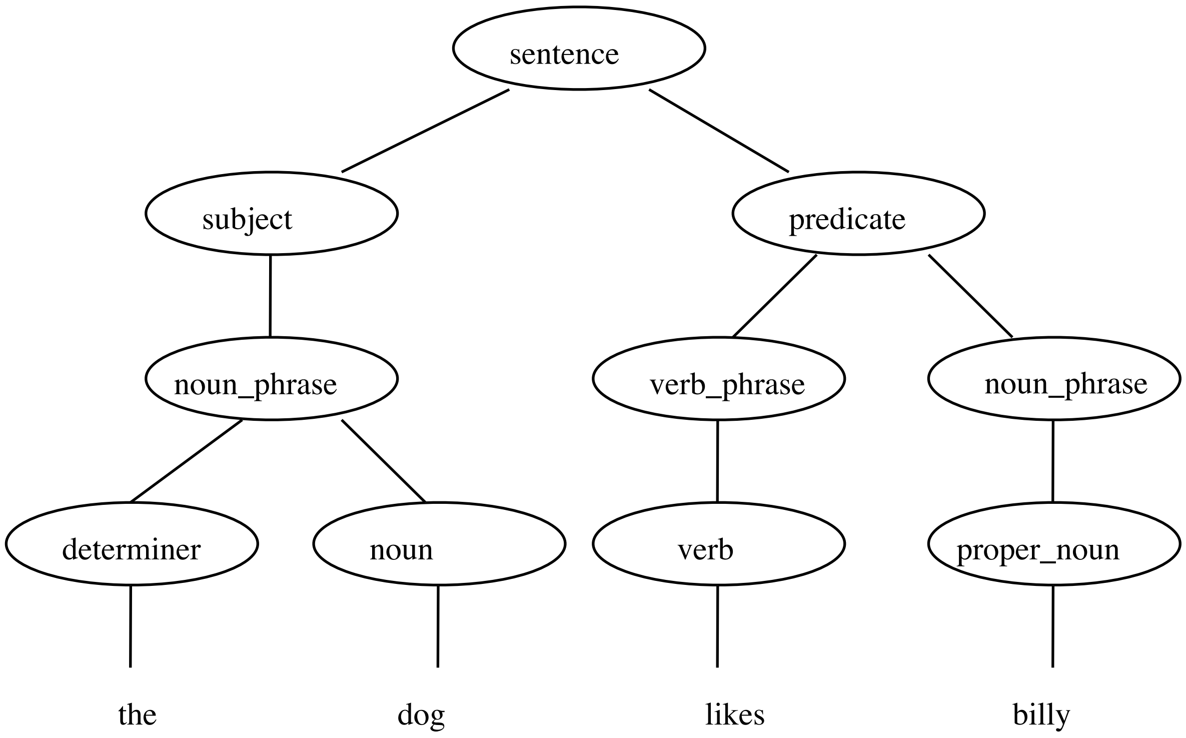
```
sentence --> subject, predicate.
subject --> noun_phrase.
noun_phrase --> determiner, noun.
noun_phrase --> proper_noun.
proper_noun  -->  [ billy ].
determiner --> [the].
noun --> [dog].
predicate --> verb_phrase
predicate --> verb_phrase, noun_phrase.
verb-phrase --> verb
verb --> [ran].
verb --> [likes].
```

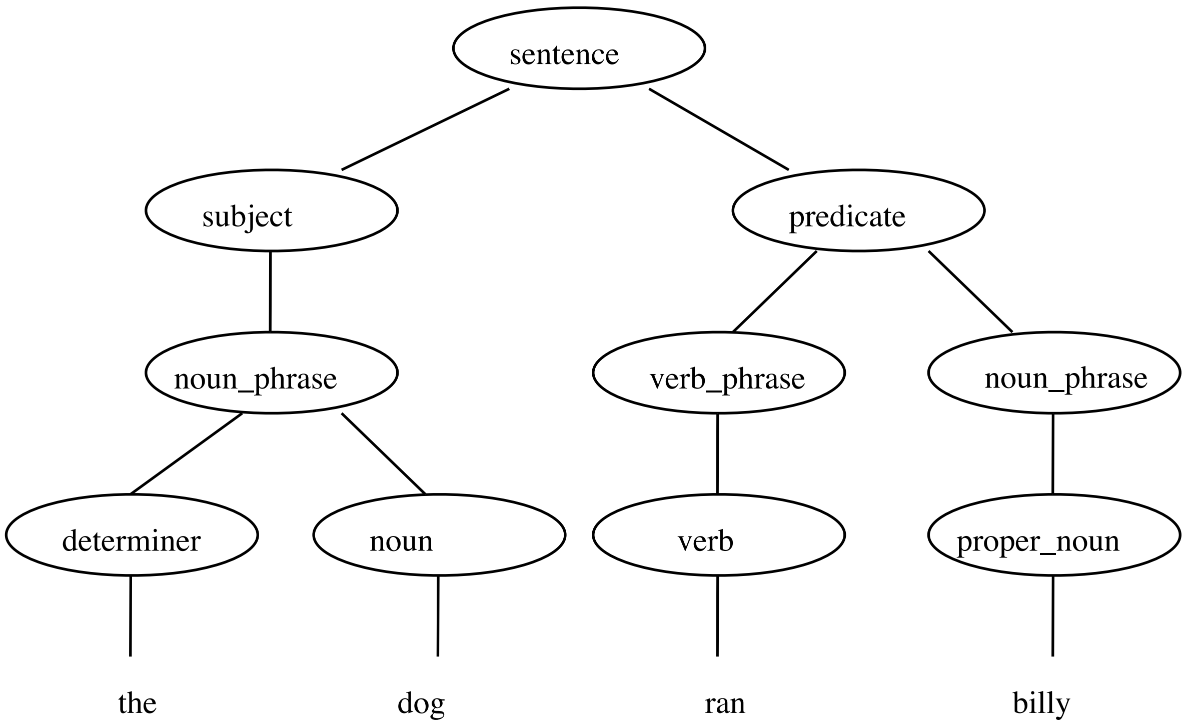
   b) The following sentences - among many others - could be generated using this grammar:

      billy ran
      the dog likes billy
      (but also) the dog ran billy ... (A more sophisticated grammar might preclude this by distinguishing between transitive and intransitive verbs)

c) These sentences have parse trees as follows







6. In a DCG, it is possible to have <u>recursive</u> or self-referential rules.

   a) For example, most languages (including English) allow adjective phrases composed of multiple adjectives. For example, in English we might say c
   "the big ball" (one adjective)
   "the big blue ball" (two adjectives)
   "the big shiny blue ball" (three adjectives)
   ...

   b) In a DCG, a list containing an arbitrary length list of elements (including possibly 0) can be specified like this:

   ```
   adjectives --> [].
   adjectives --> adjective, adjectives.
   ```

   If we want to require at least one adjective to make an adjective phrase, we can then add the following:

   ```
   adjective_phrase --> adjective, adjectives.
   ```

c) When writing a recursive rule, one must avoid left-recursion. Left recursion arises when the first symbol on the right-hand side is the same as the symbol on the left-hand side.

   E.g. the following would **not** work::

```
adjectives --> adjectives, adjective.
```

   The reason is that, in applying the rule for adjectives, Prolog would first look for an occurrence of adjectives, which would result in again applying the rule for adjectives, which would result in looking for an occurrence of adjectives - sort of like one of those pictures where a person is looking at a mirror with a mirror behind, so that one gets an infinite series of reflections of the person looking in the mirror ...

7. In a DCG, the non-terminal symbols often have arguments. Arguments may be used to specify context information or to construct a symbolic representation (parse tree), or both.

   a) Example: arguments for number.

```
sentence --> noun_phrase(Number),
             verb_phrase(Number),
             noun_phrase(_).
```

   PROJECT - this specifies that the noun phrase and the verb phrase for a sentence must agree in number (singular or plural). But the number of the direct object is irrelevant.

   b) Example: arguments for parse tree.

```
sentence(sentence(subject(Subject), verb(Verb),
         object(Object))) -->
    noun-phrase(Subject),
    verb-phrase(Verb),
    noun-phrase(Object).
```

   PROJECT

c) Example: combining the two.

```
sentence(sentence(subject(Subject),
                  verb(Verb),
                  object(Object))) -->
     noun-phrase(Subject, Number),
          verb-phrase(Verb, Number),
          noun-phrase(Object, _).
```

PROJECT

Note: It is important to be consistent about the order of arguments. In the case of noun_phrase, and verb_phrase the "number" argument comes first and then the parse tree argument.

In the case of sentence, we don't need to specify a "number" argument - that is, consistency is necessary for each non-terminal. It would be legal to put the "number" argument first for a noun_phrase and last for a verb_phrase - though this would be very confusing and is not recommended!

8. Sometimes, a DCG rule needs a more sophisticated test than simple pattern-matching.

   Example: In English, we use "an" as the indefinite article before words beginning with a vowel, and "a" as the indefinite article before other words We might represent this by the following pair of rules:

```
noun_phrase(singular, indefinite(N)) -->
    [ an ], noun(singular, N), { begins_with_vowel(N) }.
noun_phrase(singular, indefinite(N)) -->
    [ a ], noun(singular, N),{not begins_with_vowel(N) }.
```

   PROJECT

Where the “begins_with_vowel” test would be written using facilities of Prolog we don’t need to discuss here. The important thing is the use of braces to set off “pure Prolog” code as opposed to grammar rules.

9. Of course. a DCG needs to make use of a vocabulary, or list of known words.

   a) The simplest way to handle this is with a series of grammar rules whose right-hand side is a terminal symbol.

      Example: PROJECT

```
noun(singular, dog) --> [ dog ].
noun(plural, dog) --> [ dogs ].
proper_noun(billy) --> [ billy ].
verb(singular, present, like) --> [ likes ].
verb(plural, present, like) --> [ like ].
verb(_, past, like) --> [ liked ].
verb(_, future, like) --> [ will, like ].
```

      Note: In the case of noun and verb the parse tree uses the “root” form of the word.

   b) More sophisticated vocabularies might handle issues of morphology - i.e. the vocabulary might record a single entry for each root form, with additional code handling issues like formation of plurals, verb tense ...

      This gets tricky, though, as the following English examples illustrate. (Other languages have similar issues)

      (1) The plural of house is houses, but the plural of mouse is mice.

      (2) The plural of goose is geese, but the plural of moose is moose.

      ...

10. Once again, be reminded of the fact that the examples we are developing here are simple ones using Prolog DCG rules. "Real" systems use more sophisticated techniques based on similar principles.

    Here is a complete but simple grammar, with a limited vocabulary:

    PROJECT , Walk through Example Grammar

C. When a grammar has been specified by a DCG, Prolog can use it to <u>parse</u> a sentence - that is, to find the parse tree that could have been used to produce a sentence.

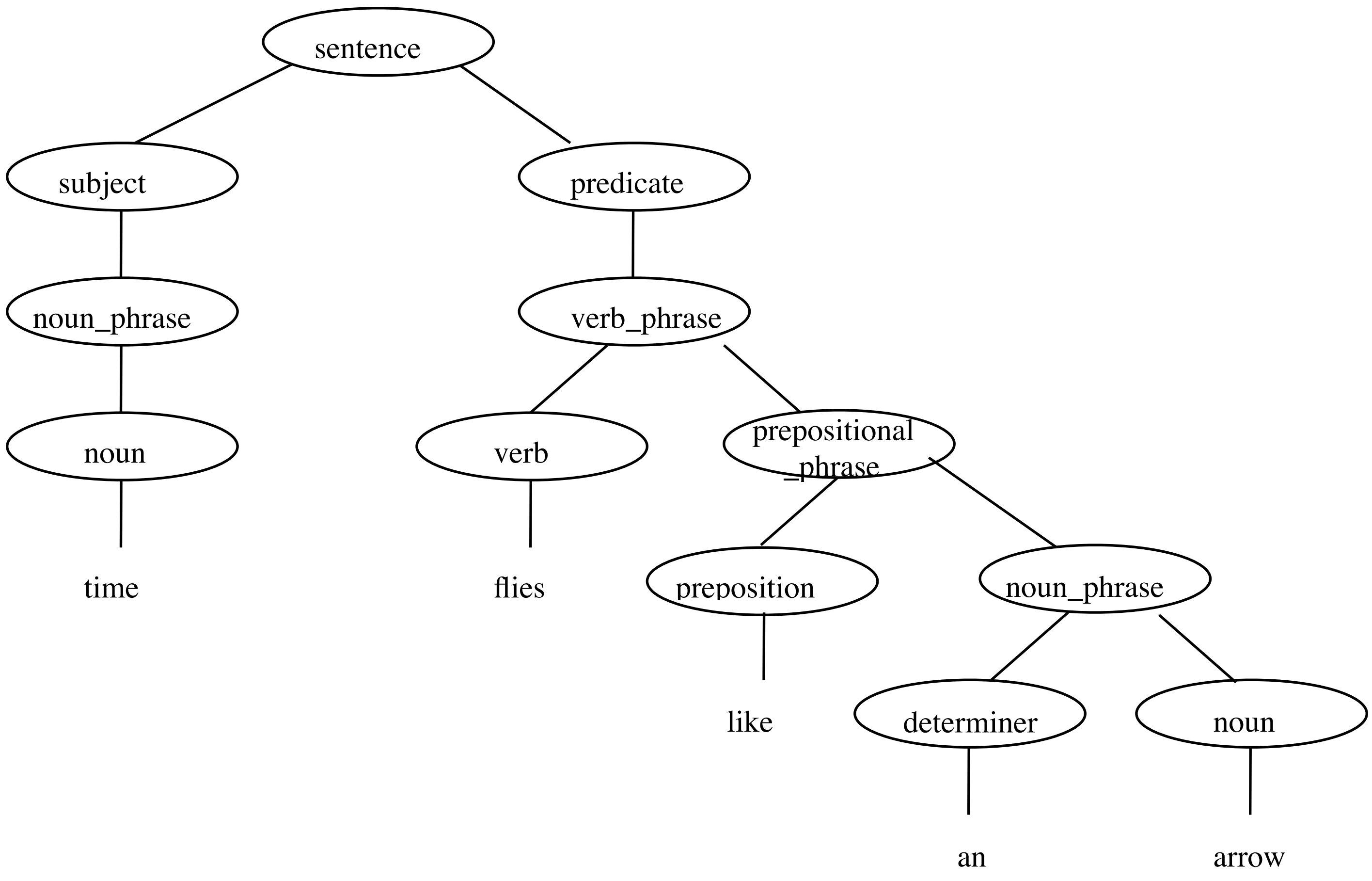
   The strategy Prolog is not particularly efficient, so while this works for simple demonstrations, one would not use this approach directly in a "real" application.

   DEMO Natural Language Parse Tree Program

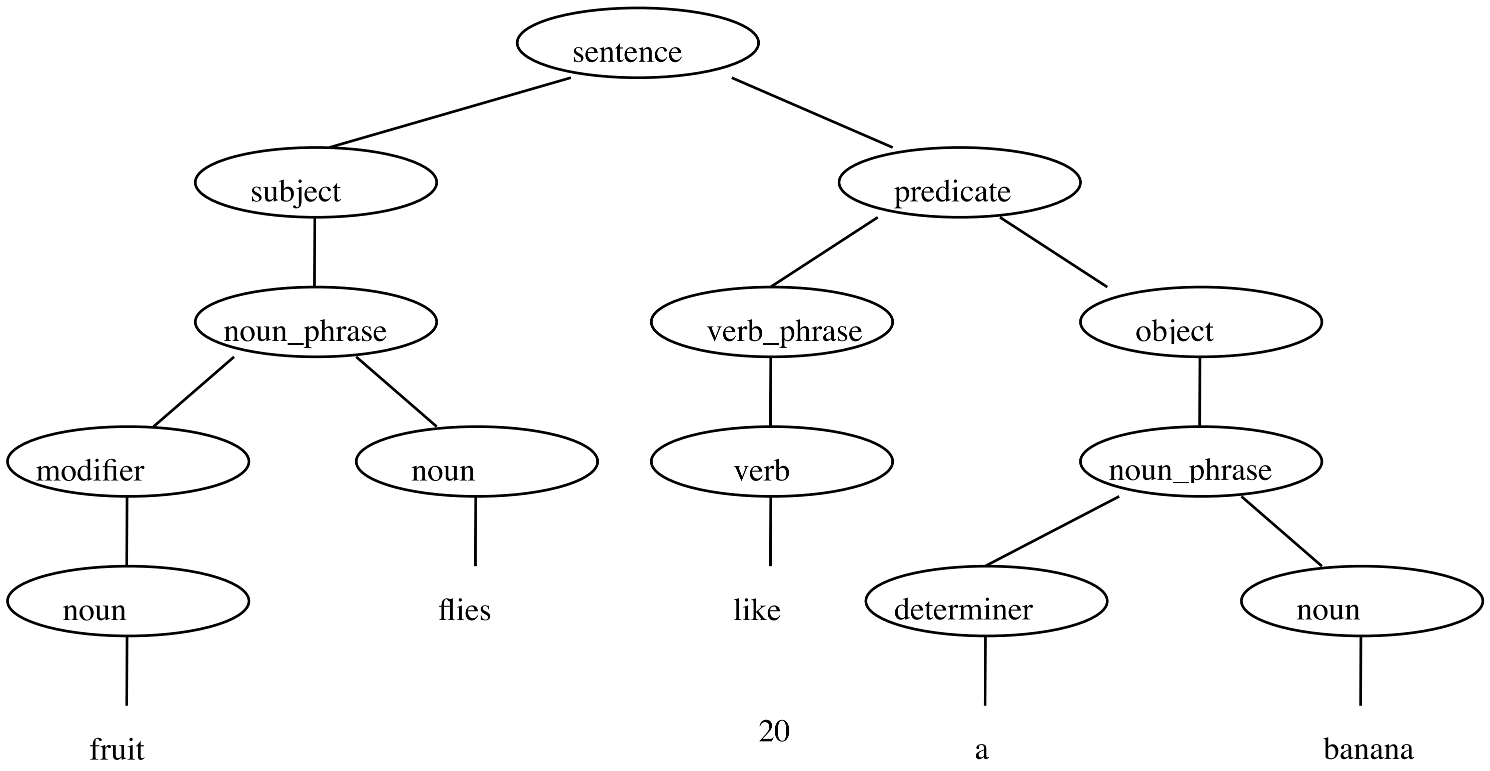
   HANDOUT Grammar

D. Earlier, we looked at a few cases where either semantic or pragmatic analysis would be needed to determine the correct parse. In such cases, the different interpretations actually will be manifested as different parse trees.

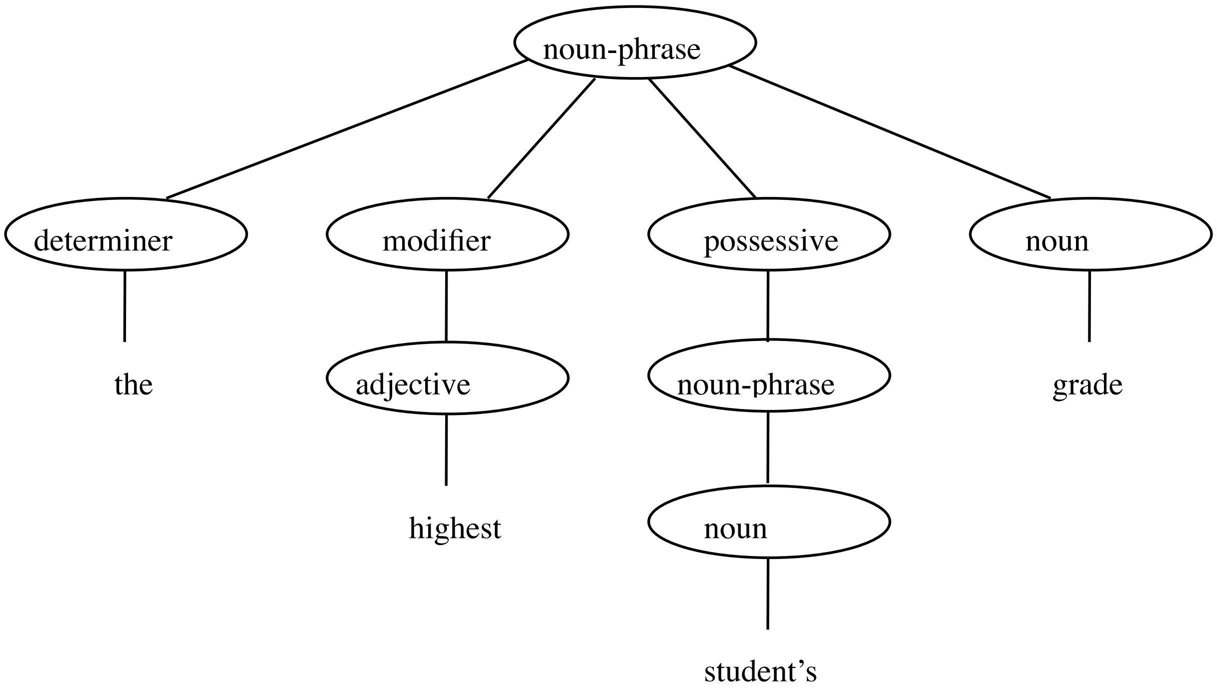
1. “Time flies like an arrow”



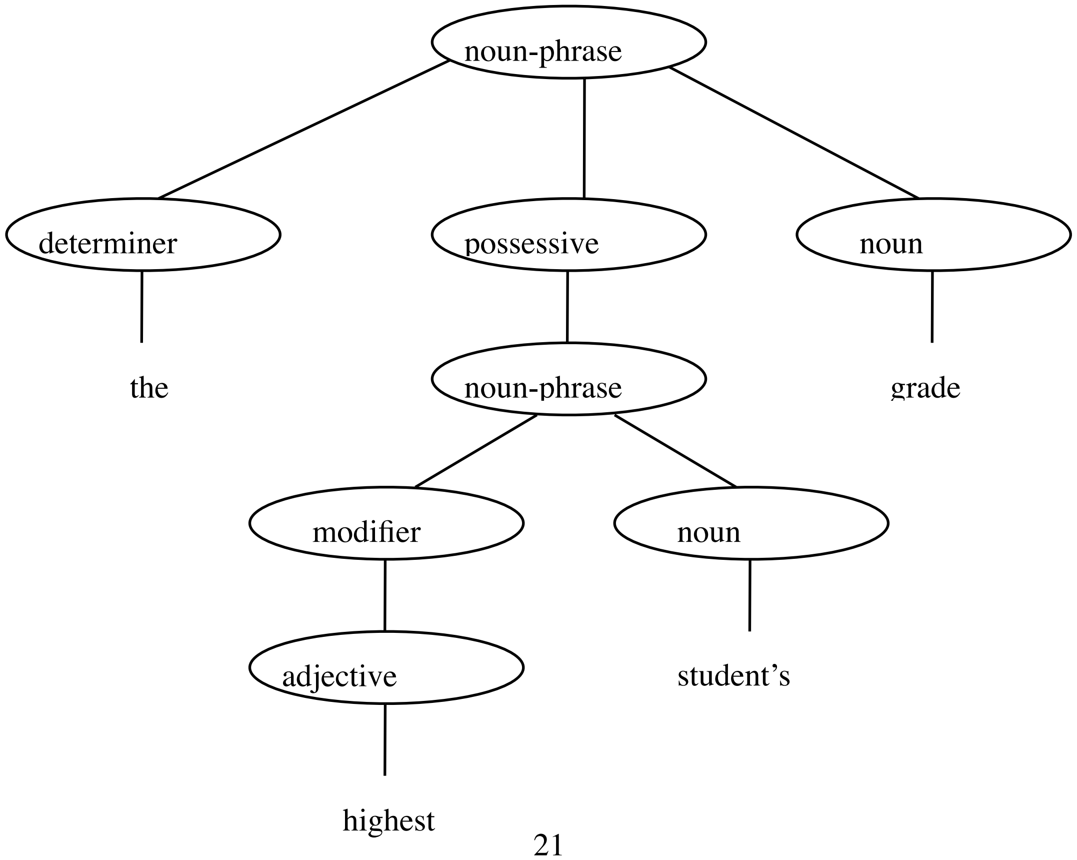
2. “Fruit flies like a banana”



3. “The highest student’s grade” - “highest” modifies “grade”



4. “The highest student’s grade” - “highest” modifies “student”



IV. **Semantics**

A. Semantics is concerned with <u>meaning</u>.

B. Before we discuss this, though, we want to talk a bit about a key question - what does it mean to "understand" something?

1. One writer offers the following definition

   "A system *understands* when it takes the actions that the user intended. This definition is "operational" ... There are times, though, when this operational definition of "understanding" ... does not seem adequate ... A system that accumulates knowledge without having a chance to apply it may or may not be understanding ... For such situations it can be useful to have a definition of understanding in terms of the *internal* behavior of the system, rather than the external behavior ... A system *understands* some input when it creates an appropriate conceptual structure, makes appropriate changes to a conceptual structure, or makes an appropriate modification to its knowledge base." (Tanimoto, *The Elements of Artificial Intelligence* p. 353)

   What do you think about this definition?

   ASK

2. Of course, it was this very matter of "understanding" that gave rise to the article by Searle that you read. Searle's article was prompted by the claim that an early (and very successful) natural language AI system could "understand" stories - which it certainly did in terms of Tanimoto's definition.

   a) How do you think Searle would define "understand"?

      ASK

b) We will discuss the article itself shortly

C. AI typically uses compositional semantics - which basically means that the meaning of a whole sentence is a composition of the meanings of its words - e.g.

The meaning of “the dog likes billy” is composed from the meanings of “the dog”, “likes” and “billy”

D. As we have already noted, semantics can also influence parsing. For example, for the sentences “time flies like an arrow” and “fruit flies like a banana”, it is conceivable that a parser (given an appropriate grammar) might produce two parse trees in each case, with semantic analysis rejecting one in each case.

1. There is no way to compose the meanings of “time” as a modifier and “flies” as a noun.

2. The composition of “fruit” as a noun with “flies” as a verb, though possible, is not probable.

E. In symbolic AI, the notion of “meaning” is defined in terms of the relationship between individual symbols and the world, and an operational definition like that given by Tanimoto is, in practice, what is used.

Some examples

1. blocks world planner NL

a) Demonstrate

b) Project, go over grammar Note how the grammar translates English input into a goal that the planner can construct a plan for

2. isa hierarchy NL

a) Demonstrate

b) Project, go over grammar.

(1) Note how the grammar translates user input into an internal form as an assertion or question, which the engine then handles.

(2) Note how the engine then runs the grammar backwards to produce an English rendition of its response.

V. Pragmatics

A. One of the key issues in pragmatics has to do with the intention of a speech act. What a sentence means literally, and what it really means, may be two very different things.

1. Example: Recall our discussion of the sentence “Can you tell me the time?”

2. Another example: In a movie, if a mafia figure says to a store owner “the last three people who didn’t buy my services ended up having their stores burned”, the statement is formally a statement of fact but is clearly meant to be understood as a threat.

3. Cawsey gives a number of other illustrations

4. Speech act theory is a field within philosophy which addresses issues like this. (Interestingly, Searle is one of the key figures in this field.)

B. Another issue has to do with recognizing the antecedents of pronouns.

1. In the simplest case, a pronoun refers to the nearest noun.

   Example: "John loves steak. Last night he went to Outback Steak House."

2. Of course,many times this is not the case. But often there are clues like gender which make the correct interpretation clear.

   Example: In the sentences: "John gave Mary flowers. He loves her", it's clear that the first pronoun ("he") in the second sentence refers to the first noun in the first sentence ("John"), and the second pronoun ("her") refers to the second noun ("Mary").

   But in the sentences "John gave Mary flowers. She can't stand him" the first pronoun refers to the second noun and vice versa.

3. Of course, sometimes there is ambiguity. Consider the following:

   "John left a small tip for the waiter. He was really unhappy".
   Is the second sentence saying that John left a small tip because he was unhappy with the service, or is it saying that the waiter was unhappy because John game him a small tip?

VI.Language Generation

A. We have focussed on language recognition but - as has already been noted - a formal grammar can be "run backwards" to generate a sentence from a parse tree.

1. Example: DEMO with simple grammar looked at earlier

   ```
   phrase(sentence(sentence(subject(definite(dog)),
                            predicate(present(like)),
                            object(billy))),
          S).
   ```

2. The natural language isa program uses this.

   DEMO again

   Show the "restate" rule in the code

B. In fact, a natural language translation system is possible in principle by using two grammars - one for the source language and one for the target language.

1. The input sentence is translated into a parse tree using the first grammar.

2. Then the parse tree is translated into the target language using the second grammar.
3. This approach has not been particularly successful, however.

   Early system - English to Russian and then Russian back to English translated "The spirit is willing but the flesh is weak" into "The vodka is strong but the mean is rotten"!

4. Interestingly, the Google translation facilities don't actually work this way. They use a statistical technique based on having a very large corpus of documents in two languages. (They started with official UN documents and official UN languages). Translation is based on correlation between phrases occurring in one language and translation in another.

I.e. there is no attempt made to “understand” or even parse the text being translated! But, interestingly, this approach to language translation (which has been used elsewhere as well as by Google) has proved to be the most effective.

VII. **Discussion of Searle Article**

A. What specific claims (of Schank) did Searle object to?

ASK

B. What was the essence of his objection?

ASK

C. What replies have others made to Searle’s point? - What were the replies, and how did Searle answer them?

ASK

D. Did you think any of the replies to Searle are actually be valid?

ASK

E. What does Searle say about whether a machine can think?

ASK

Do you agree?