We want to communicate with computers using natural language (spoken and written).

- unstructured natural language — allow any statements, but make mistakes or failure.
- controlled natural language — only allow unambiguous statements that can be interpreted (e.g., in supermarkets or for doctors).

There is a vast amount of information in natural language.

Understanding language to extract information or answering questions is more difficult than getting extracting gestalt properties such as topic, or choosing a help page.

Many of the problems of AI are explicit in natural language understanding. “AI complete”.

© D. Poole and A. Mackworth 2016

Artificial Intelligence, Lecture 13.4, Page 1
Syntax describes the form of language (using a grammar).
Semantics provides the meaning of language.
Pragmatics explains the purpose or the use of language (how utterances relate to the world).

Examples:
This lecture is about natural language.
The green frogs sleep soundly.
Colorless green ideas sleep furiously.
Furiously sleep ideas green colorless.
Beyond N-grams

- A man with a big hairy cat drank the cold milk.
- Who or what drank the milk?
Beyond N-grams

- *A man with a big hairy cat drank the cold milk.*
- Who or what drank the milk?

Simple syntax diagram:

```
s
   np  vp
  /    \
 /     \
np  pp  np
   \
   \
   a  man  drank
  /    \
 /     \
with np
   \
   \
   a  big  hairy  cat
```

©D. Poole and A. Mackworth 2016  Artificial Intelligence, Lecture 13.4, Page 4
Context-free grammar

- A **terminal symbol** is a word (perhaps including punctuation).
- A **non-terminal symbol** can be rewritten as a sequence of terminal and non-terminal symbols, e.g.,

  \[
  \text{sentence} \rightarrow \text{noun_phrase, verb_phrase}
  \]

  \[
  \text{verb_phrase} \rightarrow \text{verb, noun_phrase}
  \]

  \[
  \text{verb} \rightarrow \text{[drank]}
  \]

- Can be written as a logic program, where a sentence is a sequence of words:

  \[
  \text{sentence}(S) \leftarrow \text{noun_phrase}(N), \text{verb_phrase}(V), \text{append}(N, V, S).
  \]

  To say word “drank” is a verb:

  \[
  \text{verb}([\text{drank}]).
  \]
**Difference Lists**

- Non-terminal symbol $s$ becomes a predicate with two arguments, $s(T_1, T_2)$, meaning:
  - $T_2$ is an ending of the list $T_1$
  - all of the words in $T_1$ before $T_2$ form a sequence of words of the category $s$.

- Lists $T_1$ and $T_2$ together form a **difference list**.

- “the student” is a noun phrase:

  
  
  \[
  \text{noun\_phrase}([\text{the}, \text{student}, \text{passed}, \text{the}, \text{course}],
  [\text{passed}, \text{the}, \text{course}])
  \]
Difference Lists

- Non-terminal symbol $s$ becomes a predicate with two arguments, $s(T_1, T_2)$, meaning:
  - $T_2$ is an ending of the list $T_1$
  - all of the words in $T_1$ before $T_2$ form a sequence of words of the category $s$.

- Lists $T_1$ and $T_2$ together form a difference list.

- “the student” is a noun phrase:

  ```
  noun_phrase([the, student, passed, the, course],
  [passed, the, course])
  ```

- The word “drank” is a verb:

  ```
  verb([drank|W], W).
  ```
The grammar rule

\[ \text{sentence} \leftarrow \text{noun\_phrase}, \text{verb\_phrase} \]

means that there is a sentence between \( T_0 \) and \( T_2 \) if there is a noun phrase between \( T_0 \) and \( T_1 \) and a verb phrase between \( T_1 \) and \( T_2 \):

\[
\text{sentence}(T_0, T_2) \leftarrow \\
\text{noun\_phrase}(T_0, T_1) \land \\
\text{verb\_phrase}(T_1, T_2).
\]
The rewriting rule

\[ h \mapsto b_1, b_2, \ldots, b_n \]

says that \( h \) is \( b_1 \) then \( b_2 \), \ldots, then \( b_n \):

\[
h(T_0, T_n) \leftarrow b_1(T_0, T_1) \land b_2(T_1, T_2) \land \ldots \land b_n(T_{n-1}, T_n).
\]

using the interpretation

\[
\begin{array}{c}
T_0 \quad T_1 \quad T_2 \cdots T_{n-1} \quad T_n \\
\hline
b_1 \quad b_2 \quad \quad \quad \quad \quad \quad b_n
\end{array}
\]
Non-terminal $h$ gets mapped to the terminal symbols, $t_1, \ldots, t_n$:

$$h([t_1, \cdots, t_n|T], T)$$

using the interpretation

Thus, $h(T_1, T_2)$ is true if $T_1 = [t_1, \ldots, t_n|T_2]$. 
see
http://artint.info/code/Prolog/ch12/cfg_simple.pl

What will the following query return?

\texttt{noun\_phrase([the, student, passed, the, course, with, a, computer], R)}. 
see
http://artint.info/code/Prolog/ch12/cfg_simple.pl

What will the following query return?

\[\text{noun\_phrase}([\text{the, student, passed, the, course, with, a, computer}], R)\].

How many answers does the following query have?

\[\text{sentence}([\text{the, student, passed, the, course, with, a, computer}], R)\].
Two mechanisms can make the grammar more expressive: extra arguments to the non-terminal symbols arbitrary conditions on the rules.

We have a Turing-complete programming language at our disposal!
Add an extra argument representing a parse tree:

\[
\text{sentence}(T_0, T_2, s(NP, VP)) \leftarrow \\
\text{noun\_phrase}(T_0, T_1, NP) \land \\
\text{verb\_phrase}(T_1, T_2, VP).
\]
Enforcing Constraints

Add an argument representing the number (singular or plural), as well as the parse tree:

\[
\text{sentence}(T_0, T_2, \text{Num}, s(NP, VP)) \leftarrow \\
\text{noun\_phrase}(T_0, T_1, \text{Num}, NP) \land \\
\text{verb\_phrase}(T_1, T_2, \text{Num}, VP). \\
\]

The parse tree can return the determiner (definite or indefinite), number, modifiers (adjectives) and any prepositional phrase:

\[
\text{noun\_phrase}(T, T, \text{Num}, no\_np). \\
\text{noun\_phrase}(T_0, T_4, \text{Num}, np(Det, Num, Mods, Noun, PP)) \leftarrow \\
\text{det}(T_0, T_1, \text{Num}, Det) \land \\
\text{modifiers}(T_1, T_2, Mods) \land \\
\text{noun}(T_2, T_3, \text{Num}, Noun) \land \\
\text{pp}(T_3, T_4, PP). \\
\]
see

http://artint.info/code/Prolog/ch12/nl_numbera.pl
How can we get from natural language to a query or to logical statements?

Goal: map natural language to a query that can be asked of a knowledge base.

Add arguments representing the individual and the relations about that individual. E.g.,

\[ \text{noun\_phrase}(T_0, T_1, O, C_0, C_1) \]

means

- \( T_0 - T_1 \) is a difference list forming a noun phrase.
- The noun phrase refers to the individual \( O \).
- \( C_0 \) is list of previous relations.
- \( C_1 \) is \( C_0 \) together with the relations on individual \( O \) given by the noun phrase.
see

http://artint.info/code/Prolog/ch12/nl_interface.pl
The student took many courses. Two computer science courses and one mathematics course were particularly difficult. The mathematics course...
The student took many courses. Two computer science courses and one mathematics course were particularly difficult. The mathematics course... 

Who was the captain of the Titanic? 
Was she tall?