Speech Acts

Speech acts achieve the speaker's goals:
- Inform: ``There's a pit in front of you''
- Query: ``Can you see the gold?''
- Command: ``Pick it up''
- Promise: ``I'll share the gold with you''
- Acknowledge: ``OK''

Speech act planning requires knowledge of
- Situation
- Semantic and syntactic conventions
- Hearer's goals, knowledge base, and rationality

Stages in communication (informing)
- Intention: S wants to inform H that P
- Generation: S selects words W to express P in context C
- Synthesis: S utters words W
- Perception: H perceives W in context C'
- Analysis: H infers possible meanings P1, ... Pn
- Disambiguation: H infers intended meaning Pi
- Incorporation: H incorporates Pi into KB

How could this go wrong?
- Insincerity: S doesn't believe P
- Speech wreck ignition failure
- Ambiguous utterance
- Differing understanding of current context C<>C'
**Stages in communication (informing)**

<table>
<thead>
<tr>
<th>Intention:</th>
<th>Generation:</th>
<th>Synthesis:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Know($H$,”Alive/Wumpus,5,0”)</td>
<td>“The wumpus is dead”</td>
<td>[known or unknown]</td>
</tr>
</tbody>
</table>

**Perception:**

- “The wumpus is dead”

**Analysis:**

- NP
- VP
- Article: The
- Noun: wumpus
- Verb: is
- Adjective: dead

**Disambiguation:**

- $S$ $\rightarrow$ $NP$ $VP$
- Article $\rightarrow$ the | a | an | ...

**Incorporation:**

- $\Box$
- TELL KB
- “Alive/Wumpus,5,0”

**Grammar Types**

- **Regular:** nonterminal $\rightarrow$ terminal | nonterminal
  
  $S \rightarrow aS$
  
  $S \rightarrow \Lambda$

- **Context-free:** nonterminal $\rightarrow$ anything
  
  $S \rightarrow aSB$

- **Context-sensitive:** more nonterminals on right-hand side
  
  $ASB \rightarrow AAaBB$

**Recessively enumerable:** no constraints

**Natural language probably context-free**

**Speech Acts**

- Animals use isolated symbols for sentences
  
  - restricted set of communicable propositions, no generative capacity

- **Grammar** specifies the compositional structure of complex message (speech, language, music)
  
  - A formal language is a set of strings of terminal symbols
  
  - Each string in the language can be analyzed/generated by the grammar

- **The grammar is a set of rewrite rules,**
  
  - e.g., $S$ $\rightarrow$ $NP$ $VP$
  
  - Article $\rightarrow$ the | a | an | ...

- Here $S$ is the sentence symbol, $NP$ and $VP$ are nonterminals, and the, a ... terminals

**The Wumpus Lexicon**

- **Noun** $\rightarrow$ stench | breeze | glitter | nothing | wumpus | pit | pits | gold | east | ...

- **Verb** $\rightarrow$ is | see | smell | shoot | feel | stinks | go | grab | carry | kill | turn | ...

- **Adjective** $\rightarrow$ right | left | east | south | back | smelly | ...

- **Adverb** $\rightarrow$ here | there | nearby | ahead | right | left | east | south | back | ...

- **Pronoun** $\rightarrow$ me | you | I | it | S/HE | Y’ALL...

- **Name** $\rightarrow$ John | Mary | Boston | UCB | PAJC | ...

- **Article** $\rightarrow$ the | a | an | ...

- **Preposition** $\rightarrow$ to | in | on | near | ...

- **Conjunction** $\rightarrow$ and | or | but | ...

- **Digit** $\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

**Divided into open and closed classes**
The Wumpus Grammar

\[
S \rightarrow NP \ VP \\
| S \ Conjunction \ S \\
\]

1 + feel a breeze

I feel a breeze + and + I smell a v

\[
NP \rightarrow \text{Pronoun} \\
| \text{Noun} \\
| \text{Article Noun} \\
| \text{Digit Digit} \\
| NP \ PP \\
| NP \ RelClause \\
\]

pits

the + wumpus

3 4

the wumpus + to the east

the wumpus + that is smelly

\[
VP \rightarrow \text{Verb} \\
| VP \ NP \\
| VP \ Adjective \\
| VP \ PP \\
| VP \ Adverb \\
\]

stinks

feel + a breeze

is + smelly

turn + to the east

go + ahead

\[
PP \rightarrow \text{Preposition} \ NP \\
 light Clause \rightarrow \text{that} \ VP \\
\]

to + the east

that + is smelly

Grammatical Judgements

Formal language $L_1$ may well differ from natural language $L_2$

Adjusting is a learning problem

Real grammars typically 10-500 pages, insufficient for proper English

Syntax in NLP

Most view syntactic structure as an essential step towards meaning

``Mary hit John'' <> ``John hit Mary''

``And since I was not informed---as a matter of fact, since I did not know that there were excess funds until we, ourselves, in that checkup after the whole thing blew up, and that was, if you'll remember, that was the incident in which the attorney general came to me and told me that he had seen a memo that indicated that there were no more funds.''

Parse Trees

Exhibit the grammatical structure of a sentence

``I shoot the wumpus''
**Parsing CFGs**

Many different parsing algorithms
- Top-down
- Bottom-up
- Chart-parsing (aka CYK algorithm)
  - Dynamic programming
  - $O(n^3)$

```
[S: [NP:[Pronoun:I]]
 [VP:[VP:[Verb:shoot]]
  [NP:[Article:the][Noun:wumpus]]])
```

---

**Top-down parsing**

- Initial state: $[S: ?]$  
- Successor function:
  - Select rules for leftmost node in tree with unknown children and apply
  - $[S:[S:?][Conjunction:?][S:?]]$
  - $[S:[NP:?][VP:?]]$
- Goal Test: 
  - Check whether leaves of the parse tree correspond to (complete) input

Problem for Top-down: left-recursive rules $S \rightarrow S$  
Conjunction $S$  
infinite loops

---

**Top-down parsing**

Successful top-down parse
```
[S: ?]
[S:[NP:?][VP:?]]
[S:[NP:[Art:?][Noun:?]][VP:?]]
[S:[NP:[Art:the][Noun:wumpus]][VP:?]]
[S:[NP:[Art:the][Noun:wumpus]][VP:[Verb:?]]]
[S:[NP:[Art:the][Noun:wumpus]][VP:[Verb:stinks]]]
```

Alternative notation

| S       | [The, wumpus, stinks] |
| NP VP   | [The, wumpus, stinks] |
| Art Noun VP | [The, wumpus, stinks] |
| Noun VP | [wumpus, stinks] |
| VP      | [stinks] |
| Verb    | [stinks] |
| []      | [] |

---

**Bottom up parsing**

- Initial state 
  - $[\text{the, wumpus, is, dead}]$
- Successor function 
  - If subsequence at pos i matches right-hand of rule then replace by left hand 
  - $[[\text{Art:the}], \text{wumpus, is, dead}]$
- Goal state: 
  - A single state with root $S$

Problem for Bottom up: 
Art $\rightarrow []$
Bottom up

- [the, wumpus, is, dead]
- [[Art:the], wumpus, is, dead]
- [[Art:the][Noun:wumpus], is, dead]
- [[NP:[Art:the][Noun:wumpus]], is, dead]
- [[NP:[Art:the][Noun:wumpus]], [Verb:is], dead]
- [[NP:[Art:the][Noun:wumpus]], [VP:[Verb:is]], dead]
- [[NP:[Art:the][Noun:wumpus]], [VP:[Verb:is]], [Adj:dead]]
- [[NP:[Art:the][Noun:wumpus]], [VP:[VP:[Verb:is]]], [Adj:dead]]
- [S:[[NP:[Art:the][Noun:wumpus]], [VP:[VP:[Verb:is]]], [Adj:dead]]]]

Bottom up parsing

<table>
<thead>
<tr>
<th>step</th>
<th>list of nodes</th>
<th>subsequence</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>the wumpus is dead</td>
<td>the</td>
<td>Article \rightarrow the</td>
</tr>
<tr>
<td>2</td>
<td>Article wumpus is dead</td>
<td>wumpus</td>
<td>Noun \rightarrow wumpus</td>
</tr>
<tr>
<td>3</td>
<td>Article Noun is dead</td>
<td>Article Noun</td>
<td>NP \rightarrow Article Noun</td>
</tr>
<tr>
<td>4</td>
<td>NP is dead</td>
<td>is</td>
<td>Verb \rightarrow is</td>
</tr>
<tr>
<td>5</td>
<td>NP Verb dead</td>
<td>dead</td>
<td>Adjective \rightarrow dead</td>
</tr>
<tr>
<td>6</td>
<td>NP Verb Adjective</td>
<td>Verb</td>
<td>VP \rightarrow Verb</td>
</tr>
<tr>
<td>7</td>
<td>NP VP Adjective</td>
<td>VP Adjective</td>
<td>VP \rightarrow VP Adjective</td>
</tr>
<tr>
<td>8</td>
<td>NP VP</td>
<td>VP</td>
<td>S \rightarrow NP VP</td>
</tr>
</tbody>
</table>

Chart parsing

- Have the students in B.Sc. Informatik take the exam of AI.
- Have the students in B.Sc. Informatik taken the exam of AI?
- Double work
  - Dynamic programming - combine bottom-up and top-down
- Chart:
  - N+1 vertices
  - Labeled edges, e.g., [0, 2 S -> NP *VP]
    - Denotes that from 0 to 2 we have a NP, and if we find a VP, from 2 to k then we have an S from 0 to k

Chart parsing

- Initialization
  - Add edge [0, 0 S' \rightarrow * S]
  - Call scanner on all words
- Add edge [i, j A \rightarrow b * c]
  - if c is empty then call extender on edge
  - else call predictor on edge
- Predictor [i, j A \rightarrow b C E]
  - With (all) rules C \rightarrow d
  - Add edge [i, j A \rightarrow b * d E]
- Extender [j, k B \rightarrow c *]
  - With (all) edges [i, j A \rightarrow d * B e]
  - Add edge [i, k A \rightarrow c * e]
- Scanner [j, k A \rightarrow b * D c]
  - Word of type D at position k
  - Add edge [j, k+1 A \rightarrow b D * c]
### Chart Parse Trace

<table>
<thead>
<tr>
<th>Edge</th>
<th>Procedure</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>INITIALIZER</td>
<td>[0, 0, S' -&gt; • S]</td>
</tr>
<tr>
<td>b</td>
<td>PREDICTOR(a)</td>
<td>[0, 0, S -&gt; • NP VP]</td>
</tr>
<tr>
<td>c</td>
<td>PREDICTOR(b)</td>
<td>[0, 0, NP -&gt; • Pronoun]</td>
</tr>
<tr>
<td>d</td>
<td>SCANNER(c)</td>
<td>[0, 1, NP -&gt; Pronoun •]</td>
</tr>
<tr>
<td>e</td>
<td>EXTENDER(b,d)</td>
<td>[0, 1, S -&gt; NP • VP]</td>
</tr>
<tr>
<td>f</td>
<td>PREDICTOR(e)</td>
<td>[1, 1, VP -&gt; • Verb]</td>
</tr>
<tr>
<td>g</td>
<td>PREDICTOR(e)</td>
<td>[1, 1, VP -&gt; • VP NP]</td>
</tr>
<tr>
<td>h</td>
<td>SCANNER(f)</td>
<td>[1, 2, VP -&gt; • Verb]</td>
</tr>
<tr>
<td>i</td>
<td>EXTENDER(g,h)</td>
<td>[1, 2, VP -&gt; • VP NP]</td>
</tr>
<tr>
<td>j</td>
<td>PREDICTOR(g)</td>
<td>[2, 2, NP -&gt; • Pronoun]</td>
</tr>
<tr>
<td>k</td>
<td>SCANNER(j)</td>
<td>[2, 3, NP -&gt; Pronoun •]</td>
</tr>
<tr>
<td>l</td>
<td>EXTENDER(i,k)</td>
<td>[2, 3, NP -&gt; VP NP •]</td>
</tr>
<tr>
<td>m</td>
<td>EXTENDER(c,j)</td>
<td>[0, 3, S -&gt; NP VP •]</td>
</tr>
</tbody>
</table>

Figure 22.9 Trace of a parse of "I feel it." For each edge a-m, we show the procedure used to derive the edge from other edges already in the chart. Some edges were omitted for brevity.

### Definite Clause Grammars

- **A form of unification based grammar**
- Nonterminals become atomic expressions
  - s(Num) -> np(Num), vp(Num)
  - Num is a variable
  - Binds to singular and plural
- **Employ unification during parsing.**
- **Directly executable in Prolog.**
- Rules directly translate to (definite) clause logic
- **Next slides employ Prolog notation.**
  - Running examples with e.g. YAP prolog or SWI prolog
Non-terminals with arguments

- sentence: \(\rightarrow\) noun_phrase(N), verb_phrase(N).
- noun Phrase: \(\rightarrow\) article(N), noun(N).
- verb_phrase(N): \(\rightarrow\) intransitive verb(N).
- article(singular): \(\rightarrow\) [a].
- article(singular): \(\rightarrow\) [the].
- article(plural): \(\rightarrow\) [the].
- noun(singular): \(\rightarrow\) [turtle].
- noun(plural): \(\rightarrow\) [turtles].
- intransitive verb(singular): \(\rightarrow\) [sleeps].
- intransitive verb(plural): \(\rightarrow\) [sleep].

Case Marking

- pronoun(singular, nominative): \(\rightarrow\) [he]; [she]
- pronoun(singular, accusative): \(\rightarrow\) [him]; [her]
- pronoun(plural, nominative): \(\rightarrow\) [they]
- pronoun(plural, accusative): \(\rightarrow\) [them]

- sentence: \(\rightarrow\) np(Number, nominative), vp(Number)
- vp(Number): \(\rightarrow\) v(Number), np(X, accusative)
- np(Number, Case): \(\rightarrow\) pronoun(Number, Case)
- np(Number, X): \(\rightarrow\) det, n(Number)

He sees her. She sees him. They see her.
But not Them see he.

Principles DCG Parsing

- Given
  - u, l1, ..., ln (goal)
  - t -> t1, ..., tm (renamed)
- Generate
  - (t1, ..., tm, l1, l2, ..., ln) \(\Theta\)
  - Where \(\Theta = \text{mgu}(u, t)\)

Top down parsing with DCG

- sentence: \[He,sees,her\]
- np(Number, nom), vp(Number): \[He,sees,her\]
- pronoun(Number, nom), vp(Number): \[He,sees,her\]
- \(\Theta = \{\text{Num} = \text{sing}\}\)
  - pronoun(sing, nom): \[he\]
- vp(sing): \[sees,her\]
- \(\Theta' = \{\text{Number} = \text{sing}\}\)
  - vp(Number): \[v(Number), \text{np(X, accusative)}\]
- v(sing), np(X, accusative): \[sees,her\]
  - np(X, accusative): \[her\]
- \(\Theta'' = \{X = \text{sing}\}\)
  - np(sing, accusative): \[her\]
  - pronoun(sing, accusative): \[her\]
  - []
Subcategorization

\[
\begin{align*}
\text{vp} & \rightarrow v(1). & v(1) & \rightarrow \text{[sleep]} \\
\text{vp} & \rightarrow v(2), \text{np}. & v(2) & \rightarrow \text{[chase]} \\
\text{vp} & \rightarrow v(3), \text{np}, \text{np}. & \vdots \\
\text{vp} & \rightarrow v(4), s. \\
\end{align*}
\]

Verb | Complement | Example
--- | --- | ---
Sleep | None | The cat slept
Chase | One NP | The cat chased the dog
Give | Two NP | John gave Bill the book
Say | sentence | John said he loved Mary

Constructing parse trees

\[
\begin{align*}
\text{sentence}(s(NP, VP)) & \quad \rightarrow \text{noun\_phrase}(NP), \text{verb\_phrase}(VP). \\
\text{noun\_phrase}(np(N)) & \quad \rightarrow \text{proper\_noun}(N). \\
\text{noun\_phrase}(np(Art, Adj, N)) & \quad \rightarrow \text{article}(Art), \text{adjective}(Adj), \text{noun}(N). \\
\text{noun\_phrase}(np(Art, N)) & \quad \rightarrow \text{article}(Art), \text{noun}(N). \\
\text{verb\_phrase}(vp(TV)) & \quad \rightarrow \text{intransitive\_verb}(TV). \\
\text{verb\_phrase}(vp(TV, NP)) & \quad \rightarrow \text{transitive\_verb}(TV), \text{noun\_phrase}(NP). \\
\text{article}(art(tha)) & \quad \rightarrow \text{[the]}. \\
\text{adjective}(adj(lazy)) & \quad \rightarrow \text{[lazy]}. \\
\text{adjective}(adj(rapid)) & \quad \rightarrow \text{[rapid]}. \\
\text{proper\_noun}(pn(achilles)) & \quad \rightarrow \text{[achilles]}. \\
\text{noun}(n(turtle)) & \quad \rightarrow \text{[turtle]}. \\
\text{intransitive\_verb}(iv(sleeps)) & \quad \rightarrow \text{[sleeps]}. \\
\text{transitive\_verb}(tv(beats)) & \quad \rightarrow \text{[beats]}. \\
\end{align*}
\]

Semantic Interpretation

\[
\begin{align*}
\text{np}(\text{fido}) & \rightarrow \text{[fido]}. \\
\text{np}(\text{felix}) & \rightarrow \text{[felix]}. \\
\text{v}(X \uparrow \text{slept}(X)) & \rightarrow \text{[slept]}. \\
\text{v}(Y \uparrow (X \uparrow \text{chased}(X, Y))) & \rightarrow \text{[chased]}. \\
\text{s}(\text{Pred}) & \rightarrow \text{np}(\text{Subj}), \text{vp}(\text{Subj}^*\text{Pred}). \\
\text{vp}(\text{Subj}^*\text{Pred}) & \rightarrow \text{v}(\text{Subj}^*\text{Pred}). \\
\text{vp}(\text{Subj}^*\text{Pred}) & \rightarrow \text{v}(\text{Obj}^*(\text{Subj}^*\text{Pred})), \text{np}(\text{Obj}). \\
\end{align*}
\]

Lambda Expressions

\[
\begin{align*}
X \uparrow \text{slept}(X) & \text{ stands for } \lambda X. \text{ slept}(X). \\
Y \uparrow (X \uparrow \text{chased}(X, Y)) & \text{ stands for } \lambda X. \lambda Y. \text{ chased}(X, Y). \\
\end{align*}
\]

These lambda-expressions denote relationships. They can be instantiated by unification.

With unification:

\[
\begin{align*}
X \uparrow \text{slept}(X) \text{ with } X = \text{fido} \text{ yields } \text{fido} \uparrow \text{slept}(\text{fido}). \\
\text{unify } \text{fido} \uparrow \text{slept}(\text{fido}) \text{ with } X \uparrow R \text{ yields } R = \text{slept}(\text{fido}). \\
\end{align*}
\]

In lambda-calculus:

\[
(\lambda X. \text{ slept}(X)) \text{ fido gives slept(fido)}. 
\]
Natural Language Processing

- Complex process
  - Requiring knowledge, reasoning and AI in general
- Grammars to represent natural languages
- Parsing techniques for syntactic analysis
- DCG add expressive power to CFGs
  - Convenient in practice
  - Unification based grammars