Administrivia

• Veterans Day is November 11th
  – *No class Wednesday*
Today's Topics

• 538 Presentations

• Homework 7 review
  – Grammar programming

• Writing natural language grammars
  – "learn by doing"
  – a worked example
538 Presentations

• From lecture 1 syllabus
  – 438
    • homeworks: 100%
  – 538
    • homeworks: 75%
    • (sub-)chapter presentation: 25%

If you are currently taking 438, and you plan to do the HLT Master's Program consider volunteering to do the presentation
538 Presentations

538 presentations

• *We can’t cover as many topics as we’d like this semester*

• **next semester this course continues as 581**

• Select a *topic* from some sections from the textbook
  – *not a full chapter*
  – *don’t pick something too big or too small ...*

• Range: chapter 14 through 25
  – you can’t pick the same thing as your classmate
  – First come, first served basis
538 Presentations

Your job: present the selected topic in class
• produce slides
• you’ll be evaluated on the quality of the presentation
  – don’t just simply copy the textbook!
  – **important**: you'll be evaluated on how well you communicate the essential ideas employed
  – communicate your understanding of the tradeoffs and limitations etc. be prepared to take questions from me and your fellow classmates
538 Presentations

• Scheduling
  – Topic selection: **begins now**
  – Talk length: 10 mins
  – first come, first served basis
  – **Email me** your 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} choices
  – **Email me** your 1\textsuperscript{st} and 2\textsuperscript{nd} choice of presentation dates:
    • December 7\textsuperscript{th} and 9\textsuperscript{th}
Homework 7 Review
Homework 7 Review

• Question 1:
  – Give a regular DCG grammar for the language with strings containing:
    • an odd number of a's
    • and an even number of b's (even: 0, 2, 4, ...)
    • assume alphabet is {a, b}
  – Examples:
    • aaa, abb, babaabb
    • *aab, *ab, *ε
  – Show your grammar working, i.e. it accepts and rejects positive and negative examples, respectively.
  – Allow rules of the form:
    • x --> [].
Homework 7 Review

• Central idea:
  – you don't need to do (unbounded) counting!
  – think of states

Apply this to both a's and b's
Homework 7 Review

- Example grammar:
Homework 7 Review

• Question 2:
  – Order your grammar rules so it enumerates strings in the language.
  – Show it working.
  – Is your grammar capable of enumerating all the strings of the language? Explain why or why not.
Homework 7 Review

• Enumeration:

```prolog
?- s(X, []).  
X = [a] ;
X = [a, a, a] ;
X = [a, a, a, a, a] ;
X = [a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a, a, a, a, a] ;
X = [a, a, a, a, a, a, a, a, a, a, a, a, a, a, a] ;
```

```prolog
?- s(X, []).  
X = [a] ;
X = [a, b, a, b, a] ;
X = [a, b, a, b, a, b, a, b] ;
X = [a, b, a, b, a, b, a, b, a] ;
X = [a, b, a, b, a, b, a, b, a, a] ;
X = [a, b, a, b, a, b, a, b, a, a, a] ;
X = [a, b, a, b, a, b, a, b, a, a, a, a] ;
X = [a, b, a, b, a, b, a, b, a, a, a, a, a] ;
X = [a, b, a, b, a, b, a, b, a, a, a, a, a, a] ;
```

```prolog
?- s(X, []).  
X = [b, b, a] ;
X = [b, b, a, a] ;
X = [b, b, a, a, a] ;
X = [b, b, a, a, a, a] ;
X = [b, b, a, a, a, a, a] ;
X = [b, b, a, a, a, a, a, a] ;
X = [b, b, a, a, a, a, a, a, a] ;
X = [b, b, a, a, a, a, a, a, a, a] ;
X = [b, b, a, a, a, a, a, a, a, a, a] ;
```

Homework 7 Review

• Question 3:
  – (a) Determine how many strings of size 3 are in the language? List them.
  – (b) Compute how many strings of size < 10 are there in the language.
  – Examples:
    • [b, b, a] is of size 3
    • [a, a, a, b, b, b, b] is of size 7
Homework 7 Review

- Question 3:
  - (a) Determine how many strings of size 3 are in the language? List them.

```prolog
?- s([X1,X2,X3],[]).
X1 = X2, X2 = X3, X3 = a ;
X1 = a,
X2 = X3, X3 = b ;
X1 = X2, X2 = b,
X3 = a ;
X1 = X3, X3 = b,
X2 = a.
```

```prolog
?- L = [X1,X2,X3], s(L,[]).
L = [a, a, a],
X1 = X2, X2 = X3, X3 = a ;
L = [a, b, b],
X1 = a,
X2 = X3, X3 = b ;
L = [b, b, a],
X1 = X2, X2 = b,
X3 = a ;
L = [b, a, b],
X1 = X3, X3 = b,
X2 = a.
```
Homework 7 Review

- Question 3:
  - (b) Compute how many strings of size < 10 are there in the language.

<table>
<thead>
<tr>
<th>Size</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>9</td>
<td>256</td>
</tr>
</tbody>
</table>

Total: 341
Phrase Structure Grammars

- Let's write a phrase structure grammar (PSG) using Prolog

- Mechanisms:
  1. Extra argument for Prolog term representation of a parse
  2. Extra arguments for feature value agreement
  3. Dealing with left recursive rules: grammar transformation (*Next time*)
Topics

• Mechanisms:
  1. Extra argument for Prolog term representation of a parse
  2. Extra arguments for feature value agreement
  3. Dealing with left recursive rules: grammar transformation
Let's write a simple grammar returning appropriate parse trees for sentences like:

- John kicked the ball
- the men kicked the ball
- a man kicked the ball

Stanford Parser:

- (ROOT
  - (S
    - (NP (DT the) (NNS men))
    - (VP (VBD kicked)
      - (NP (DT the) (NN ball))))
Topics

• Mechanisms:
  1. Extra argument for Prolog term representation of a parse
  2. Extra arguments for feature value agreement
  3. Dealing with left recursive rules: grammar transformation
Extra Arguments: Agreement

• **Idea:**
  – We can also use an extra argument to impose constraints between constituents within a DCG rule

• **Example:**
  – English determiner-noun number agreement
  – Data:
    • the man
    • the men
    • a man
    • *a men
  – Lexical Features:
    • *man singular
    • *men plural
Extra Arguments: Agreement

• Data:
  – the man/men
  – a man/*a men

• Grammar: (NP section)

  np(np(Y)) --> pronoun(Y).
  np(np(D,N)) --> det(D, Number),
    common_noun(N, Number).
  det(det(the), sg) --> [the].
  det(det(the), pl) --> [the].
  det(det(a), sg) --> [a].
  common_noun(n(ball), sg) --> [ball].
  common_noun(n(man), sg) --> [man].
  common_noun(n(men), pl) --> [men].
  pronoun(i) --> [i].
  pronoun(we) --> [we].

• Idea:
  give determiners a number feature as well
  and make it agree with the noun

• Rules
  • *the can combine with singular or plural nouns
  • *a can combine only with singular nouns
Extra Arguments: Agreement

- **Simplifying the grammar:**
  
  \[
  \begin{align*}
  \text{det}(&\text{det}(\text{the}), \text{sg}) \rightarrow \text{[the]}.
  \\
  \text{det}(&\text{det}(\text{the}), \text{pl}) \rightarrow \text{[the]}.
  \\
  \text{det}(&\text{det}(\text{a}), \text{sg}) \rightarrow \text{[a]}.
  \end{align*}
  \]

- **Grammar is ambiguous:**
  - two rules for determiner *the*

- **Agreement Rule (revisited):**
  - *the* can combine with singular or plural nouns
  - i.e. *the* doesn’t care about the number of the noun

- **DCG Rule:**
  
  \[
  \begin{align*}
  \text{np}(&\text{np}(D,N)) \rightarrow \text{det}(D,\text{Number}), \text{common\_noun}(N,\text{Number}).
  \\
  \text{det}(&\text{det}(\text{the}), _) \rightarrow \text{[the]}.
  \end{align*}
  \]

**Note:** _ is a variable used underscore character because we don’t care about the value of the variable
Note:

• Use of the extra argument for agreement here is basically “syntactic sugar” and lends no more expressive power to the grammar rule system

• i.e. we can enforce the agreement without the use of the extra argument at the cost of more rules

• Instead of

\[
\text{np(np}(D,N)) \rightarrow \text{det}(D, \text{Number}), \\
\quad \text{common_noun}(N, \text{Number}).
\]

we could have written:

\[
\text{np(np}(D,N)) \rightarrow \text{detsg}(D), \text{common_nounsg}(N). \\
\text{np(np}(D,N)) \rightarrow \text{detpl}(D), \text{common_nounpl}(N). \\
\text{detsg}(\text{det}(a)) \rightarrow [a]. \\
\text{detsg}(\text{det}(\text{the})) \rightarrow [\text{the}]. \\
\text{detpl}(\text{det}(\text{the})) \rightarrow [\text{the}]. \\
\text{common_nounsg}(\text{n}(\text{ball})) \rightarrow [\text{ball}]. \\
\text{common_nounsg}(\text{n}(\text{man})) \rightarrow [\text{man}]. \\
\text{common_nounpl}(\text{n}(\text{men})) \rightarrow [\text{men}].
\]
Extra Arguments: Agreement

- English exhibits subject-verb agreement

- Examples:
  - John kicked the ball
  - The men kicked the ball
  - John kicks the balls
  - The men *kicks/kick the ball

Constraint:

1. *-s form of the verb is compatible with 3rd person singular only for the subject NP
2. *uninflected* form is not compatible with 3rd person singular for the subject NP
Subject Verb Agreement

• We need feature percolation:

Subject and VP come together at this rule

<table>
<thead>
<tr>
<th>Form</th>
<th>Ending</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat</td>
<td>uninflected</td>
<td>not 3rd person singular</td>
</tr>
<tr>
<td>eats</td>
<td>-s</td>
<td>3rd person singular</td>
</tr>
<tr>
<td>ate</td>
<td>-ed</td>
<td>past</td>
</tr>
<tr>
<td>eaten</td>
<td>-en</td>
<td>past participle</td>
</tr>
<tr>
<td>eating</td>
<td>-ing</td>
<td>gerund</td>
</tr>
</tbody>
</table>

POS tags:

- VB - Verb, base form
- VBD - Verb, past tense
- VBG - Verb, gerund or present participle
- VBN - Verb, past participle
- VBP - Verb, non-3rd person singular present
- VBZ - Verb, 3rd person singular present
Subject Verb Agreement

• Implementation: using POS tags
  \[v(vb(eat),vb) \rightarrow [eat].\]
  \[v(vbd(ate),vbd) \rightarrow [ate].\]
  \[v(vbg(eating),vbg) \rightarrow [eating].\]
  \[v(vbn(eaten),vbn) \rightarrow [eaten].\]
  \[v(vbp(eat),vbp) \rightarrow [eat].\]
  \[v(vbz(eats),vbz) \rightarrow [eats].\]

• Constraint table:
  – % table of Person Number Tag possible combinations
  – table(3,plural,vb).
  – table(3,plural,vbd).
  – table(3,singular,vbz).
  – table(3,singular,vbd).

Person, Number from Subject NP
POS tag from verb