LING/C SC/PSYC 438/538
Computational Linguistics
Sandiway Fong
Lecture 9: 9/25
Administrivia

• Reminder
  – Homework 2 due today
  – to be reviewed in class on Thursday
Administrateivia

- **Last Thursday:**
  - class replaced by talk on Treebank experiments
  - *we’ll come back to statistical parsing later in the course*
Regular Grammars: Recap

- Given right recursive regular grammar
  
  \[
  \begin{align*}
  s &\rightarrow [b], \text{ ays}. \\
  \text{ays} &\rightarrow [a], \text{ a}. \\
  a &\rightarrow [a], \text{ exclam}. \\
  a &\rightarrow [a], \text{ a}. \\
  \text{exclam} &\rightarrow [!].
  \end{align*}
  \]

- Output:
  
  
  \[
  \begin{align*}
  \text{?- } s(L, []). & \quad ; \\
  L &\rightarrow [b,a,a,!] \quad ; \\
  L &\rightarrow [b,a,a,a,!] \quad ; \\
  L &\rightarrow [b,a,a,a,a,!] \quad ; \\
  L &\rightarrow [b,a,a,a,a,a,!] \quad ; \\
  L &\rightarrow [b,a,a,a,a,a,a,!] \quad ; \\
  \end{align*}
  \]

- Try
  
  \[
  \begin{align*}
  \text{?- } s(L, []). & \quad ; \\
  \end{align*}
  \]
  
  - Prolog returns one answer and waits for action
    (type ; to get the next, <return> to finish)

  Prolog is enumerating the strings of the language
Regular Grammars: Recap

- left recursive regular grammar:
  \[
  \begin{align*}
    s & \rightarrow a, ![]. \\
    a & \rightarrow \text{first}a, [a]. \\
    a & \rightarrow a, [a]. \\
    \text{first}a & \rightarrow b, [a]. \\
    b & \rightarrow [b]. 
  \end{align*}
  \]

- also works as an enumerator but not so well as a decider

- Example:
  \[
  \begin{align*}
    ?- & \ s(L,[ ]). \\
    \quad & \text{vs} \\
    ?- & \ s([b,a,!],[ ]). 
  \end{align*}
  \]

- Computational behavior
  - have to be aware of Prolog’s computation rule:
    - top-down, depth-first, left-to-right
Regular Grammars: Recap

- **Beyond regular grammars**
  - $a^n b^n = \{ab, aabb, aaabbb, aaaabbbb, \ldots \}$ $n \geq 1$
    - is not a regular language

- A regular grammar extended to allow both left and right recursive rules can accept/generate it:
  1. $a \rightarrow [a], b.$
  2. $b \rightarrow [b].$
  3. $b \rightarrow a, [b].$

- **Elementary trees**
  - generated by rules 1 + 2
  - generated by rules 1 + 3
    - *center-embedding recursive tree*
DCG

- extra parameter
  - used for generating a parse tree
- feature agreement
  - e.g. English determiner-noun agreement

- extra parameter
  - allows counting
Extra Argument and Agreement

• **Idea:**
  – We can also use an extra argument to impose constraints between constituents within a DCG rule
  – *(in fact, we can have multiple extra arguments)*

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

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

• Grammar:
  \( s(s(Y,Z)) \rightarrow \text{np}(Y), \text{vp}(Z) \).
  \( \text{np}(\text{np}(Y)) \rightarrow \text{pronoun}(Y) \).
  \( \text{pronoun}(i) \rightarrow [i] \).
  \( \text{pronoun}(\text{we}) \rightarrow [\text{we}] \).

  \( \text{np}(\text{np}(D,N)) \rightarrow \text{det}(D), \text{common_noun}(N) \).
  \( \text{det}(\text{det}(\text{the})) \rightarrow [\text{the}] \).
  \( \text{det}(\text{det}(a)) \rightarrow [a] \).
  \( \text{common_noun}(\text{n}(\text{ball})) \rightarrow [\text{ball}] \).
  \( \text{common_noun}(\text{n}(\text{man})) \rightarrow [\text{man}] \).
  \( \text{common_noun}(\text{n}(\text{men})) \rightarrow [\text{men}] \).

• Grammar contd.
  \( \text{vp}(\text{vp}(Y)) \rightarrow \text{unergative}(Y) \).
  \( \text{vp}(\text{vp}(Y,Z)) \rightarrow \text{transitive}(Y), \text{np}(Z) \).
  \( \text{unergative}(\text{v}(\text{ran})) \rightarrow [\text{ran}] \).
  \( \text{transitive}(\text{v}(\text{hit})) \rightarrow [\text{hit}] \).
Extra Argument and Agreement

• Let’s test and modify this grammar in Prolog

• Test:
  ?- s(Tree, Sentence, []).

• Sentence is a Prolog list to be supplied by user

• Tree a variable
Extra Argument and Agreement

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

• **Grammar: (NP section)**
  - np(np(Y)) --> pronoun(Y).
  - np(np(D,N)) --> det(D),
    common_noun(N,Number).
  - det(det(the)) --> [the].
  - det(det(a)) --> [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:**
  - specify singular (sg) and plural (pl) for common nouns using an extra argument

• **Rules**
  - *the* can combine with singular or plural nouns
  - *a* can combine only with singular nouns
Extra Argument and 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 Argument and Agreement

• Simplifying the grammar:
  \[ \text{det}(\text{det}(\text{the}), \text{sg}) \rightarrow [\text{the}]. \]
  \[ \text{det}(\text{det}(\text{the}), \text{pl}) \rightarrow [\text{the}]. \]
  \[ \text{det}(\text{det}(\text{a}), \text{sg}) \rightarrow [\text{a}]. \]

• Grammar is ambiguous:
  – two rules for determiner \textit{the}

• Agreement Rule (revisited):
  • \textit{the} can combine with singular or plural nouns
  • i.e. \textit{the} doesn’t care about the number of the noun

• DCG Rule:
  \[ \text{np}(\text{np}(\text{D}, \text{N})) \rightarrow \text{det}(\text{D}, \text{Number}), \text{common_noun}(\text{N}, \text{Number}). \]
  \[ \text{det}(\text{det}(\text{the}), \_ ) \rightarrow [\text{the}]. \]

\textbf{Note:} \_ is a variable
\textit{used underscore character because we don’t care about the name of the variable}
Extra Argument and Agreement

- **Note:**
  - Use of the extra argument for agreement here is basically “syntactic sugar”
  - and lends no more extra 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
  \[
  \text{np(np(D,N))} \rightarrow \text{det}(D, \text{Number}), \\
  \text{common\_noun}(N, \text{Number}).
  \]

  *we can write:*
  \[
  \text{np(np(D,N))} \rightarrow \text{detsg}(D), \\
  \text{common\_nounsg}(N). \\
  \text{np(np(D,N))} \rightarrow \text{detpl}(D), \\
  \text{common\_nounpl}(N). \\
  \text{detsg(det(a))} \rightarrow [a]. \\
  \text{detsg(det(the))} \rightarrow [the]. \\
  \text{detpl(det(the))} \rightarrow [the]. \\
  \text{common\_nounsg(n(ball))} \rightarrow [ball]. \\
  \text{common\_nounsg(n(man))} \rightarrow [man]. \\
  \text{common\_nounpl(n(men))} \rightarrow [men].
  \]
Expressive Power and Extra Arguments

• **General Question:**
  – *Does the extra argument change the system?*
  – i.e. *does it offer the grammar more expressive power?*

• **Answer:**
  – Yes

• **Example:**
  – We can write an otherwise regular-looking DCG for the non-regular language
  
    \[ L_{ab} = \{ a^n b^n \mid n \geq 1 \} \]

    *(which cannot be encoded by any regular grammar)*
Expressive Power and Extra Arguments

\[ a^+ b^+ \equiv a \text{ regular grammar} \]

\[ s(X) \rightarrow [a], b(a(X)). \]
\[ b \rightarrow [a], b. \]
\[ b \rightarrow [b], c. \]
\[ b \rightarrow [b]. \]
\[ c \rightarrow [b], c. \]
\[ c \rightarrow [b]. \]

\[ s \rightarrow [a], b. \]
\[ b \rightarrow [a], b. \]
\[ b \rightarrow [b]. \]
\[ b \rightarrow [b], c. \]
\[ b \rightarrow [b]. \]
\[ c \rightarrow [b], c. \]
\[ c \rightarrow [b]. \]

\text{regular grammar + extra argument}

Query: \(?- s(0, L, []).\)
Expressive Power and Extra Arguments

• Computation tree:
  - ?- s(0, [a, a, b, b], []).  
  - ?- b(a(0), [a, b, b], []).  
  - ?- b(a(a(0)), [b, b], []).  
  - ?- c(a(0), [b], []).  
  - Yes

\[
s(X) \rightarrow [a], b(a(X)). \\
b(X) \rightarrow [a], b(a(X)). \\
b(a(X)) \rightarrow [b], c(X). \\
b(a(0)) \rightarrow [b]. \\
c(a(X)) \rightarrow [b], c(X). \\
c(a(0)) \rightarrow [b].
\]
Class Exercise

• Implement Case = \{nom, acc\} feature agreement for subject and object positions in the example grammar

• Examples:
  – *the man hit I
  – the man hit me