Administrivia

• I’ll go through homework 1 in class next time
• You’ll receive a letter grade by email
  – *the letter grades are meant to provide you with feedback on how you’re doing...*
  – **A** means your answers are correct or only contain minor errors or variations
  – **B** means your answers are adequate but contain some non-minor errors
  – anything below a **B** means there is a problem with your understanding of the materials (review is recommended)
  – **A*** means you did especially well
Administrivia

• Next Thursday
  – *change of venue*
  – **instead of class here**, please attend the Master’s Seminar at 5pm
  – presented by the Cognitive Science Program
  – Speech, Language and Hearing Sciences Building, Rm. 205
  – I’ll be giving a research talk (accessible to non-specialists) on treebanks and statistical natural language parsers
Today’s Topics

• Prolog DCG and regular grammar recap
• more on regular grammars
  – Prolog’s computation rule
  – left and right recursive grammar rules
Chomsky Hierarchy

Type-0
DCG

Type-1
Type-2
Type-3

FSA
Regular Expressions
Regular Grammars
Definite Clause Grammars (DCG): recap

- **language**: *Sheeptalk*
  - baa!
  - baaa!
  - ba...a!

- **grammar**
  - $s \rightarrow [b], [a], a, [!]$. *(base case)*
  - $a \rightarrow [a]$. *(recursive case)*
  - $a \rightarrow a, [a]$.

Encode Sheeptalk using a regular grammar?

*This grammar is a context-free grammar (type-2) but not a regular grammar (type-3)*
Regular Grammars

• Regular or Chomsky hierarchy type-3 grammars
  – are a class of formal grammars with a restricted RHS
    • LHS \rightarrow \text{ RHS}  \quad \text{“LHS rewrites/expands to RHS”}

• format
  – \ A \rightarrow \ Bt \quad \ A \rightarrow \ t \quad \text{(left recursive)}
  or
  – \ A \rightarrow \ tB \quad \ A \rightarrow \ t \quad \text{(right recursive)}

  \quad \text{where \ A \ and \ B \ are non-terminals and \ t \ represents a terminal.}
  \ i.e. all rules contain only a single non-terminal, and (possibly) a single terminal) on the right hand side

• Note:
  – \text{can’t have both left and right recursive rules at the same time}
Regular Grammars

- In Prolog format:
  - $a \rightarrow b, [t]$. $a \rightarrow [t]$. (left recursive)
  or
  - $a \rightarrow [t], b$. $a \rightarrow [t]$. (right recursive)
  
  where $a$ and $b$ are non-terminals and $[t]$ represents a terminal.
Definite Clause Grammars (DCG): recap

- **language**: *Sheeptalk*
  - baa!
  - baaa!
  - ba...a!

- **Class exercise**:
  - Let’s give the regular grammar version of *Sheeptalk*:
    - right recursive
    - left recursive

- **context-free (but not regular) grammar**
  - s --> [b], [a], a, [!].
  - a --> [a].
    (base case)
  - a --> a, [a].
    (recursive case)
Prolog’s Computation Rule

- Top-down
- Left to right
- Depth-first

This means the order in which grammar rules are written may matter.

- infinite looping is possible unless we write the grammar carefully

every time a non-terminal is to be expanded, Prolog tries to match in the order given.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. s --&gt; [a],b.</td>
<td>P1</td>
</tr>
<tr>
<td>2. b --&gt; [a],b.</td>
<td>P2</td>
</tr>
<tr>
<td>3. b --&gt; [b],c.</td>
<td>P3</td>
</tr>
<tr>
<td>4. b --&gt; [b].</td>
<td>P4</td>
</tr>
<tr>
<td>5. c --&gt; [b],c.</td>
<td>P5</td>
</tr>
<tr>
<td>6. c --&gt; [b].</td>
<td>P6</td>
</tr>
</tbody>
</table>

backtracking is used when rules do not lead to a match.
Prolog’s Computation Rule

• Let’s trace through what Prolog does

• Command:
  • ?- trace.

• Command:
  • ?- notrace.

• Caveat:
  – grammar rules are translated into underlying Prolog rules
  – Example:
    – each rule \( x \rightarrow a, [t]. \)
    – is expanded into
      \[
      x(\text{ListBeforeX, ListAfterX}) :-
      a(\text{ListBeforeX, ListAfterA}),
      'C'(\text{ListAfterA, t, ListAfterX}).
      \]
  – This is why we make the initial call
    ? - x(\text{List, []}).
Prolog’s Computation Rule

• Prolog code:
  \[ x(\text{ListBeforeX}, \text{ListAfterX}) :- \]
  \[ a(\text{ListBeforeX}, \text{ListAfterA}), 'C'(\text{ListAfterA}, t, \text{ListAfterX}). \]

• LHS :– RHS means LHS is true if RHS is true
• , (comma) means logical conjunction
• \(x\) and \(a\) are predicates
• \(x(P_1, P_2)\) means predicate \(x\) has two parameters (or arguments) \(P_1\) and \(P_2\)
• \(x\) and \(a\) are said to have arity 2 because they take two parameters, sometimes written as \(x/2\) or \(a/2\)
• \('C'(\text{List1}, t, \text{List2})\) is a Prolog built-in predicate that is true if \(t\) is the head of the list \(\text{List1}\) and \(\text{List2}\) is the value of the rest of the list (after removing the head)
Prolog’s Computation Rule

- **Example:**
  - each rule \( x \rightarrow [t]. \)
  
    is expanded into

    \( x([t|\text{ListBeforeX}],\text{ListAfterX}). \)

- means \( x/2 \) is true
  
    if the first parameter is a list
    with head \( t \) and tail \( \text{ListAfterX} \),
    and the 2nd parameters is
    \( \text{ListAfterX} \)

- **Prolog head-tail list notation:**

  - \( [a,b,c] \)
  
    list with three elements \( a, b \) and \( c \)
  
  - can also be written as:

    - \( [a|[b|[c|[]]]] \)
  
  - generally \( [\text{head}|\text{tail}] \)

  - \text{head} = first element of the list
  
    - \( \text{tail} = \) all but the first element of the list
  
    - \( [] = \) empty list
Definite Clause Grammars (DCG)

- **language**: *Sheeptalk*
  - baa!
  - baaa!
  - ba...a!

- **Class exercise**:
  - Let’s give the regular grammar version of *Sheeptalk*:
    - right recursive
    - left recursive

- Let’s write the right recursive one first...
Definite Clause Grammars (DCG)

• **language:** *Sheeptalk*
  - baa!
  - baaa!
  - ba...a!

• **Class exercise:**
  - Let’s give the regular grammar version of *Sheeptalk*:
    - right recursive
    - left recursive

• Now the left recursive one ...

• Is there a difference between the two versions?
Definite Clause Grammars (DCG)

• To answer the question
  – try different inputs...