

For Wednesday

- Read chapter 23, sections 1-2
- FOIL exercise due

Program 5

- Any questions?

Learning mini-project

- Worth 2 homeworks
- Due Wednesday
- Foil6 is available in `/home/mecalif/public/itk340/foil`
- A manual and sample data files are there as well.
- Create a data file that will allow FOIL to learn rules for a `sister/2` relation from background relations of `parent/2`, `male/1`, and `female/1`. You can look in the `prolog` folder of my `327` folder for sample data if you like.
- Electronically submit your data file—which should be named `sister.d`, and turn in a hard copy of the rules FOIL learns.

More Examples

- Semantics

I put the plant in the window.

Ford put the plant in Mexico.

The dog is in the pen.

The ink is in the pen.

- Pragmatics

The ham sandwich wants another beer.

John thinks vanilla.

Formal Grammars

- A **grammar** is a set of **production rules** which generates a set of strings (a language) by **rewriting** the top symbol S.
- **Nonterminal** symbols are intermediate results that are not contained in strings of the language.

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$VP \rightarrow V NP$

- **Terminal** symbols are the final symbols (words) that compose the strings in the language.
- Production rules for generating words from part of speech categories constitute the lexicon.
- $N \rightarrow \text{boy}$
- $V \rightarrow \text{eat}$

Context-Free Grammars

- A context-free grammar only has productions with a single symbol on the left-hand side.
- CFG:
 $S \rightarrow NP V$
 $NP \rightarrow Det N$
 $VP \rightarrow V NP$
- not CFG:
 $AB \rightarrow C$
 $BC \rightarrow FG$

Simplified English Grammar

S -> NP VP

S -> VP

NP -> Det Adj* N

NP -> ProN

NP -> PName

VP -> V

VP -> V NP

VP -> VP PP

PP -> Prep NP

Adj* -> e

Adj* -> Adj Adj*

Lexicon:

ProN -> I; ProN -> you; ProN -> he; ProN -> she

Name -> John; Name -> Mary

Adj -> big; Adj -> little; Adj -> blue; Adj -> red

Det -> the; Det -> a; Det -> an

N -> man; N -> telescope; N -> hill; N -> saw

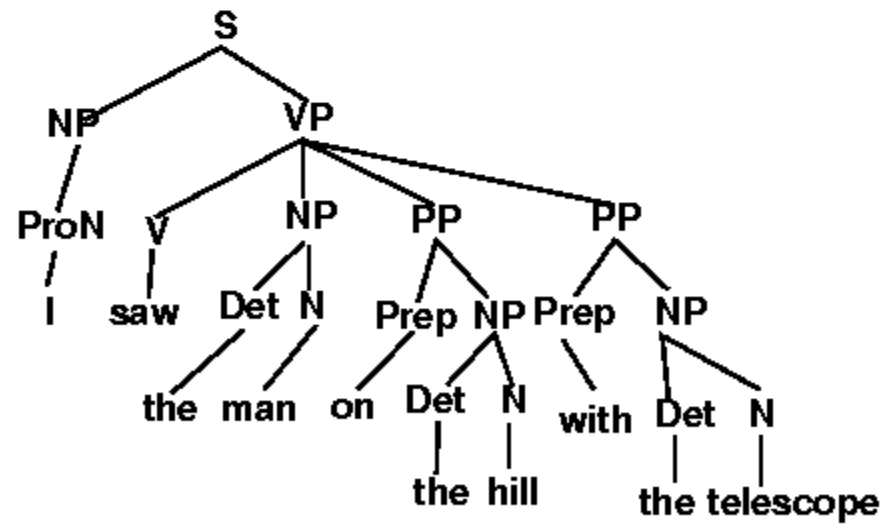
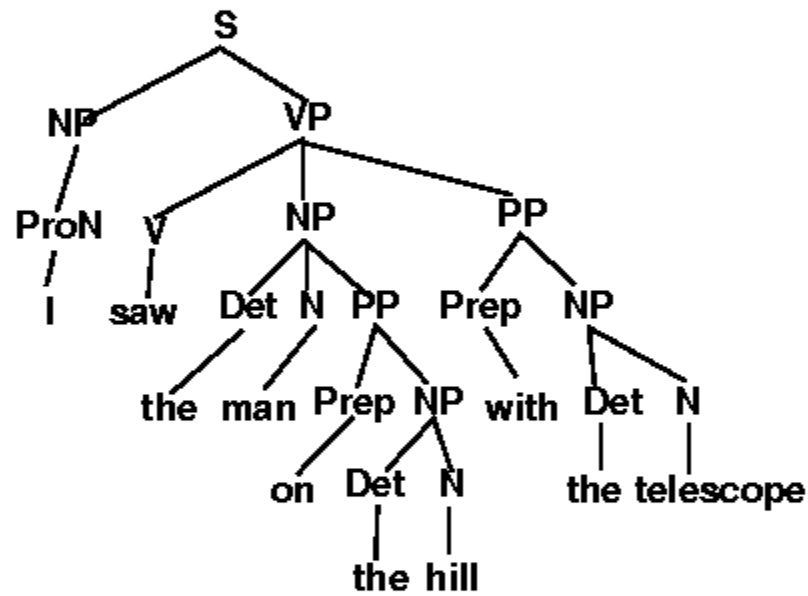
Prep -> with; Prep -> for; Prep -> of; Prep -> in

V -> hit; V -> took; V -> saw; V -> likes

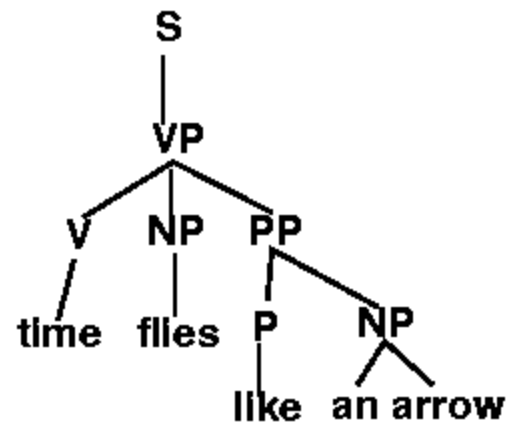
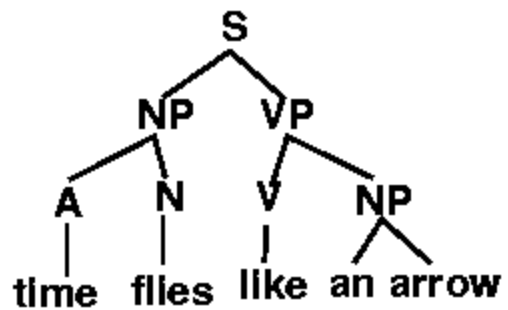
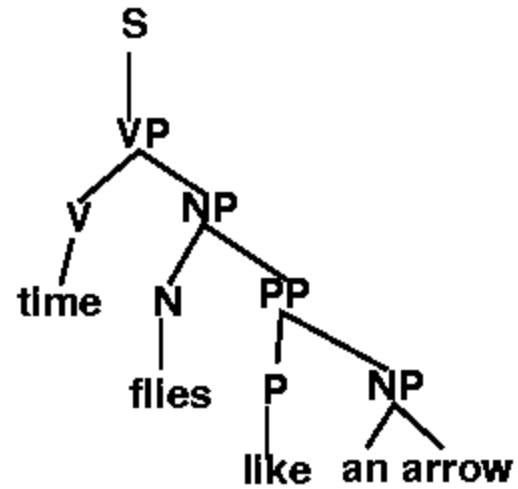
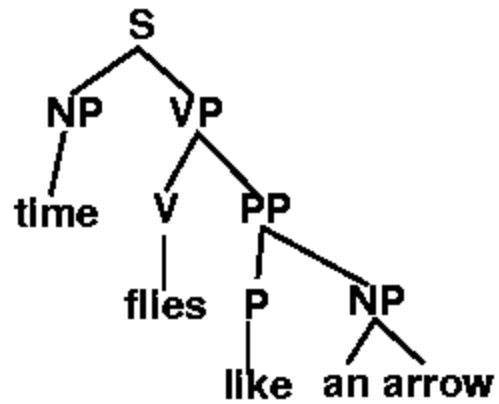
Parse Trees

- A parse tree shows the **derivation** of a sentence in the language from the start symbol to the terminal symbols.
- If a given sentence has more than one possible derivation (parse tree), it is said to be **syntactically ambiguous**.

(part 2/3), it is said to be syntactically ambiguous.



Spurious Parses



Syntactic Parsing

- Given a string of words, determine if it is grammatical, i.e. if it can be derived from a particular grammar.
- The derivation itself may also be of interest.
- Normally want to determine all possible parse trees and then use semantics and pragmatics to eliminate spurious parses and build a semantic representation.

Parsing Complexity

- **Problem:** Many sentences have many parses.
- An English sentence with n prepositional phrases at the end has at least 2^n parses.

I saw the man on the hill with a telescope on Tuesday in Austin...

- The actual number of parses is given by the Catalan numbers:

1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796...

Parsing Algorithms

- Top Down: Search the space of possible derivations of S (e.g. depth-first) for one that matches the input sentence.

I saw the man.

| | |
|------------------|------------------|
| S -> NP VP | VP -> V NP |
| NP -> Det Adj* N | V -> hit |
| Det -> the | V -> took |
| Det -> a | V -> saw |
| Det -> an | NP -> Det Adj* N |
| NP -> ProN | Det -> the |
| ProN -> I | Adj* -> e |
| | N -> man |

Parsing Algorithms (cont.)

- Bottom Up: Search upward from words finding larger and larger phrases until a sentence is found.

I saw the man.

ProN saw the man

NP saw the man

NP N the man

NP V the man

NP V Det man

NP V Det Adj* man

NP V Det Adj* N

NP V NP

NP VP

S

ProN -> I

NP -> ProN

N -> saw (dead end)

V -> saw

Det -> the

Adj* -> e

N -> man

NP -> Det Adj* N

VP -> V NP

S -> NP VP

Bottom-up Parsing Algorithm

function BOTTOM-UP-PARSE(*words*, *grammar*) **returns** a parse tree

forest \leftarrow *words*

loop do

if LENGTH(*forest*) = 1 and CATEGORY(*forest*[1]) = START(*grammar*) **then**
 return *forest*[1]

else

i \leftarrow **choose** from {1...LENGTH(*forest*)}

rule \leftarrow **choose** from RULES(*grammar*)

n \leftarrow LENGTH(RULE-RHS(*rule*))

subsequence \leftarrow SUBSEQUENCE(*forest*, *i*, *i+n-1*)

if MATCH(*subsequence*, RULE-RHS(*rule*)) **then**

forest[*i*...*i+n-1*] / [MAKE-NODE(RULE-LHS(*rule*), *subsequence*)]

else fail

end

Augmented Grammars

- Simple CFGs generally insufficient:
“The dogs bites the girl.”
- Could deal with this by adding rules.
 - What’s the problem with that approach?
- Could also “augment” the rules: add constraints to the rules that say number and person must match.

Verb Subcategorization

Semantics

- Need a semantic representation
- Need a way to translate a sentence into that representation.
- Issues:
 - Knowledge representation still a somewhat open question
 - Composition
“He kicked the bucket.”
 - Effect of syntax on semantics

Dealing with Ambiguity

- Types:
 - Lexical
 - Syntactic ambiguity
 - Modifier meanings
 - Figures of speech
 - Metonymy
 - Metaphor

Resolving Ambiguity

- Use what you know about the world, the current situation, and language to determine the most likely parse, using techniques for uncertain reasoning.

Discourse

- More text = more issues
- Reference resolution
- Ellipsis
- Coherence/focus

Survey of Some Natural Language Processing Research

Speech Recognition

- Two major approaches
 - Neural Networks
 - Hidden Markov Models
 - A statistical technique
 - Tries to determine the probability of a certain string of words producing a certain string of sounds
 - Choose the most probable string of words
- Both approaches are “learning” approaches

Syntax

- Both hand-constructed approaches and data-driven or learning approaches
- Multiple levels of processing and goals of processing
- Most active area of work in NLP (maybe the easiest because we understand syntax much better than we understand semantics and pragmatics)

POS Tagging

- Statistical approaches--based on probability of sequences of tags and of words having particular tags
- Symbolic learning approaches
 - One of these: transformation-based learning developed by Eric Brill is perhaps the best known tagger
- Approaches data-driven

Developing Parsers

- Hand-crafted grammars
- Usually some variation on CFG
- Definite Clause Grammars (DCG)
 - A variation on CFGs that allow extensions like agreement checking
 - Built-in handling of these in most Prologs
- Hand-crafted grammars follow the different types of grammars popular in linguistics
- Since linguistics hasn't produced a perfect grammar, we can't code one

Efficient Parsing

- Top down and bottom up both have issues
- Also common is chart parsing
 - Basic idea is we're going to locate and store info about every string that matches a grammar rule
- One area of research is producing more efficient parsing

Data-Driven Parsing

- PCFG - Probabilistic Context Free Grammars
- Constructed from data
- Parse by determining all parses (or many parses) and selecting the most probable
- Fairly successful, but requires a LOT of work to create the data

Applying Learning to Parsing

- Basic problem is the lack of negative examples
- Also, mapping complete string to parse seems not the right approach
- Look at the operations of the parse and learn rules for the operations, not for the complete parse at once