LING 364: Introduction to Formal Semantics

Lecture 5

January 26th

Administrivia

- Reminder:
  - Homework 1 due on tonight (midnight deadline)
  - questions? ask now
- Reading Assignment
  - Chapter 2: *Putting a Meaning Together from Pieces*

Last Time

- Translating English into logical meaning

Mary is a student

who is a student?

student(mary).

?- student(X).

Last Time

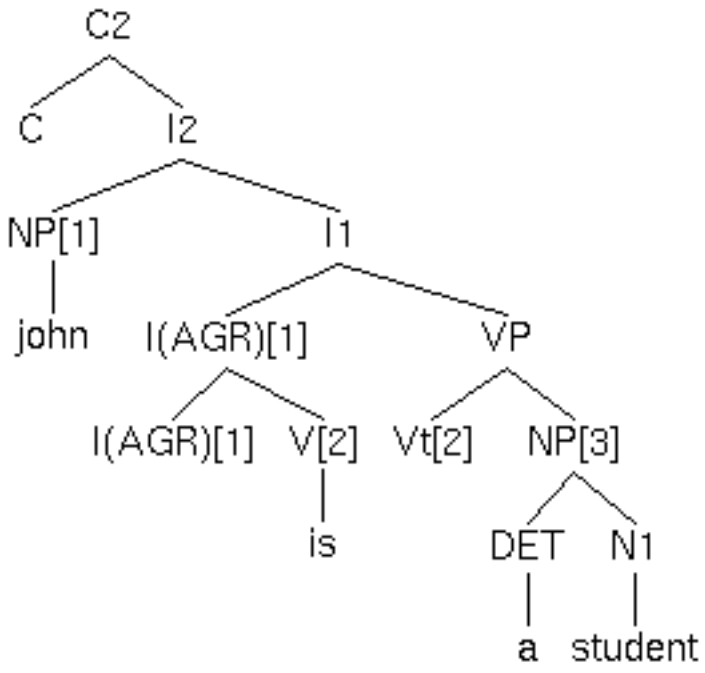
- Goal:
  - formalize language to the degree we can have systems that can understand and answer questions wrt. possible worlds

- *demo*
  - |: john is a student.
  - student(john).
  - |: mary is a student.
  - student(mary).
  - |: mary is a baseball fan.
  - baseball_fan(mary).
  - |: who is a student and not a baseball fan?
  - john.
  - | ?- go.
  - |: who is a student and a baseball fan?
  - mary.

to do this we have to be able to
(1) parse, and
(2) assign meaning to the English input

Last Time

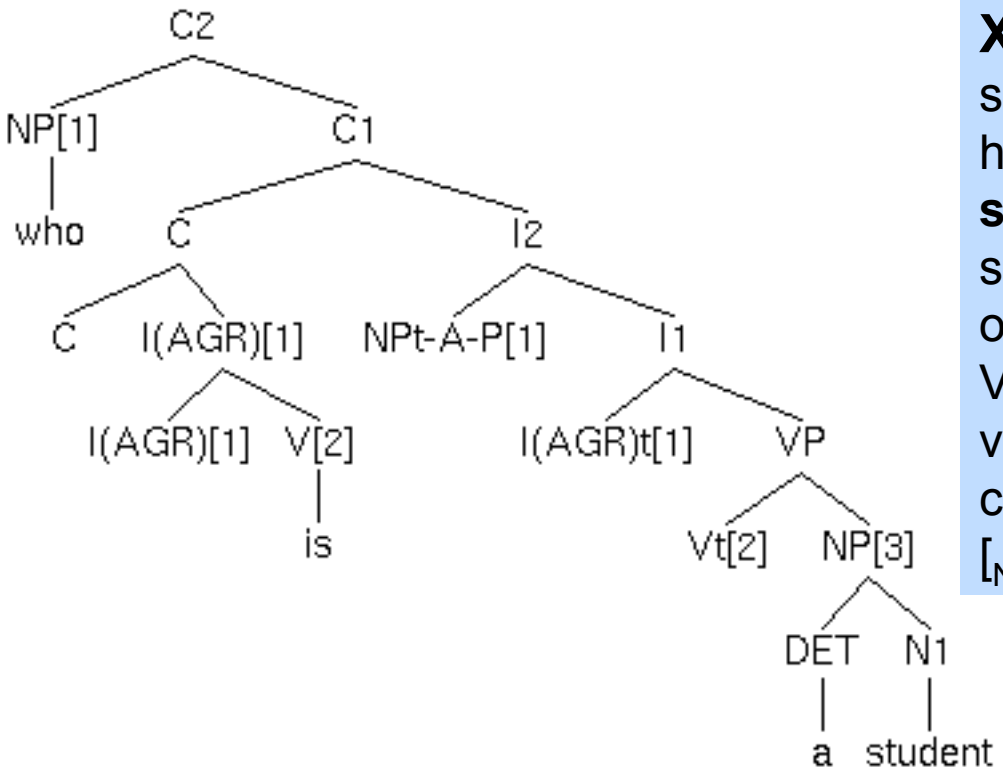
- **Syntax**:
  - A formal grammar enables us to logically break down a sentence into its constituent parts



X-bar phrase structure
subject: [$_{I2}$ [$_{NP}$ john] I1]
VP: *is a student*
copula: *is*
complement of VP:
[$_{NP}$ [$_{DET}$ a][$_{N1}$ student]]
(predicate NP)

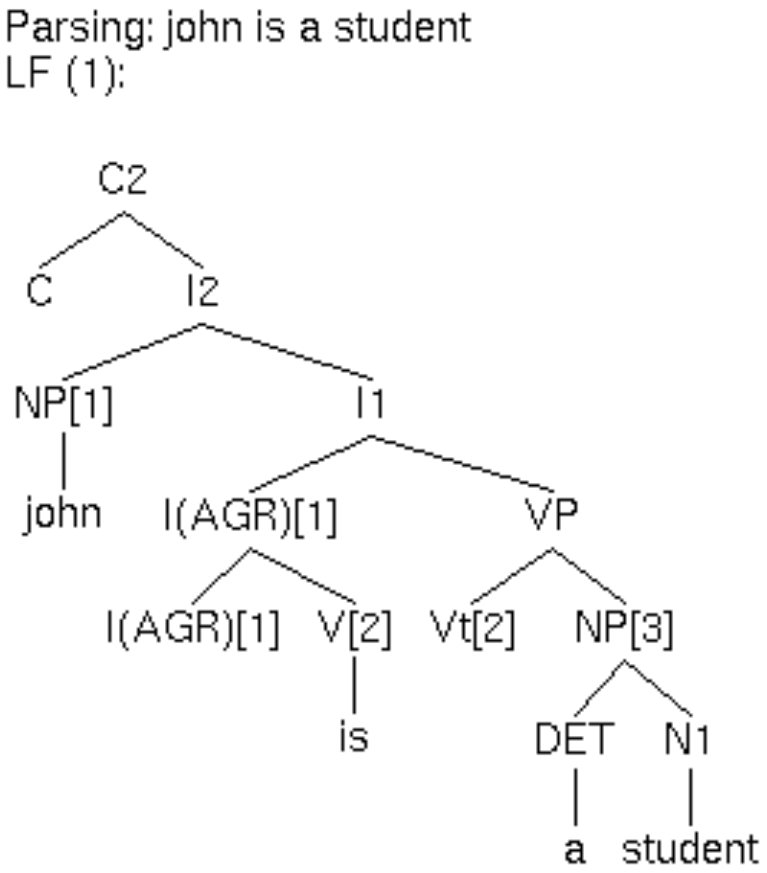
Syntactic Structure

- A formal grammar enables us to logically break down a sentence into its constituent parts



X-bar phrase structure
specifier of CP: [$_{CP}$ [$_{NP}$ who] C1]
head of CP: C: auxiliary verb *is*
subject: [$_{I2}$ [$_{NP}$ trace] I1]
subject is coindexed [1] with specifier of CP
VP: [$_{V}$trace] *a student*
verb (trace) is coindexed [2] with *is*
complement of VP:
[$_{NP}$ [$_{DET}$ a][$_{N1}$ student]]

Phrase Structure Rules



- **Simple rules:**
- SBar → S
- S → NP VP
- VP → V NP
- V → is
- NP → DET N
- NP → ProperNoun
- ProperNoun → John
- DET → a
- N → student

subject

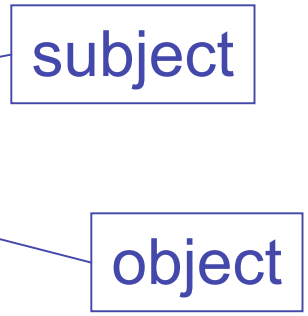
object

Phrase Structure Rules

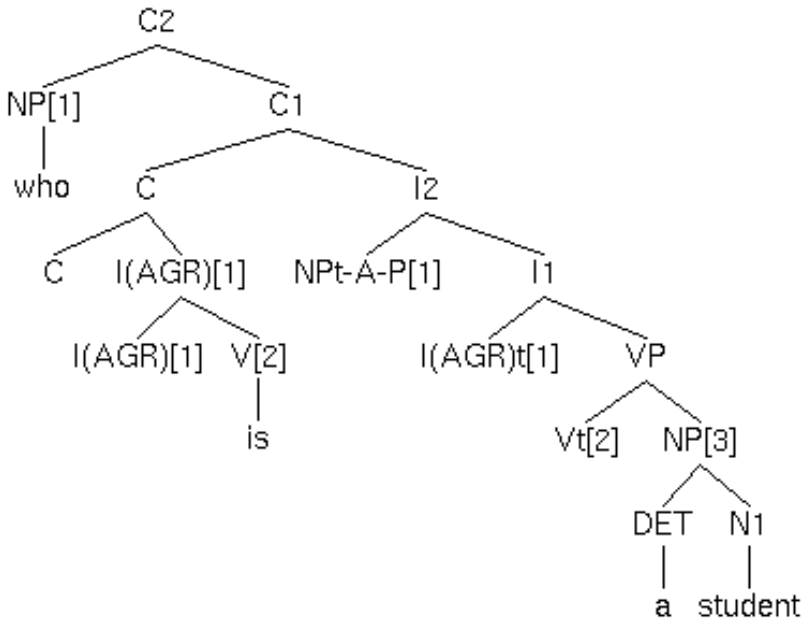
- John is a [$_{pred}$ student]
- John [$_{pred}$ likes] Mary
- John is [$_{pred}$ happy]

- which is the **predicate**?
  - V (main verb: *likes*)
  - V_{aux} *is* (copula carries little meaning)
  - complement of copula is the predicate
- Note:
  - *gotta be careful*
  - John is **the** student

- **Simple rules:**
- SBar → S
- S → NP VP
- VP → V NP
- V → is
- NP → DET N
- NP → ProperNoun
- ProperNoun → John
- DET → a
- N → student



Phrase Structure Rules



- **Rules**:
- SBar → WhNoun Aux S
- WhNoun → who
- Aux → is
- S → NPtrace VP
- NPtrace →ε
- VP → Vtrace NP
- Vtrace →ε
- NP → DET N
- DET → a
- N → student

subject

empty

object

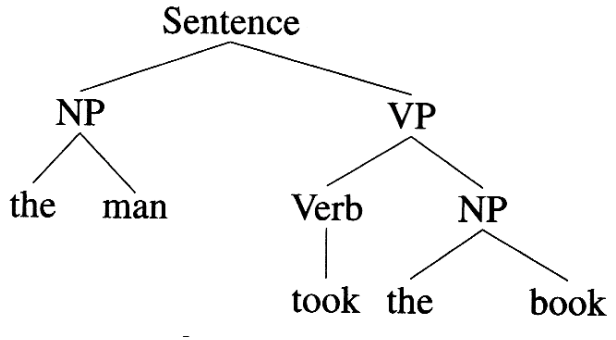
plus associations by coindexation between traces and contentful items

Today's Topics

1. What is a formal grammar?
2. Prolog's notation for formal grammars
   - **Definite Clause Grammars**
3. Discussion of Putting a Meaning Together from Pieces
4. Short Quiz

What is a formal grammar?

- **example**

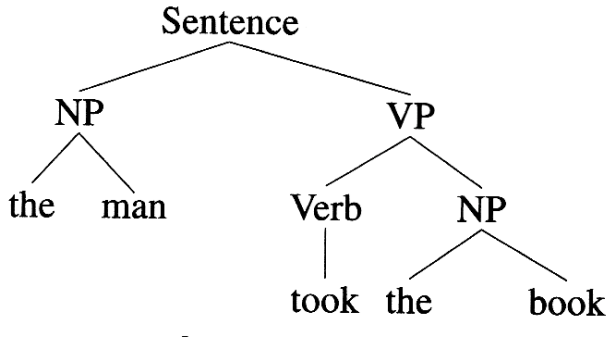


NP = Noun Phrase
VP = Verb Phrase

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book
- **production (or grammar) rule format**
  - **LHS → RHS**
    - LHS = Left Hand Side
    - → = “expands to” or “rewrites to”
    - RHS = Right Hand Side

What is a formal grammar?

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book



- **derivation**
  - **top-down (*one of many*)**

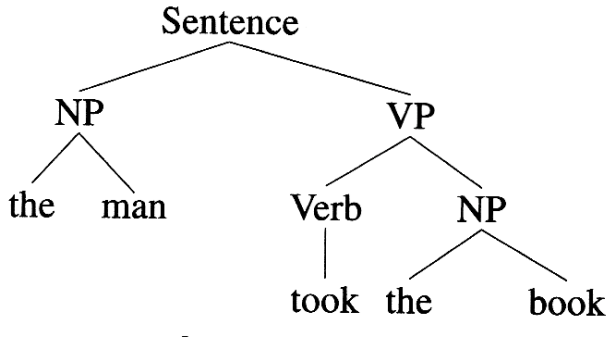
1. Sentence
2. NP VP
3. NP Verb NP
4. NP took NP
5. the man took NP
6. the man took the book

- **derivation**
  - **top-down (*alternative*)**

1. Sentence
2. NP VP
3. the man VP
4. the man Verb NP
5. the man took NP
6. the man took the book

What is a formal grammar?

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book



- **derivation**
  - **bottom-up (*one of many*)**

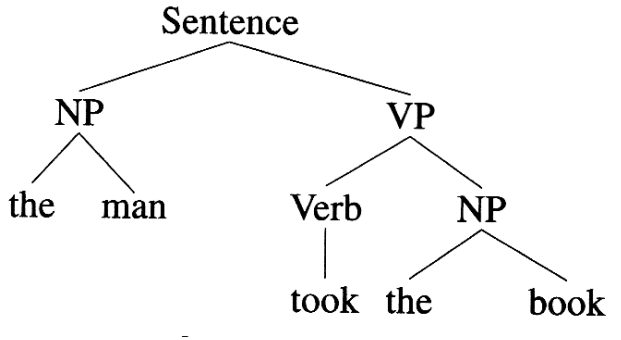
1. the man took the book
2. NP took the book
3. NP Verb the book
4. NP Verb NP
5. NP VP
6. Sentence

- **derivation**
  - **bottom-up (*alternative*)**

1. the man took the book
2. the man took NP
3. the man Verb NP
4. the man VP
5. NP VP
6. Sentence

What is a formal grammar?

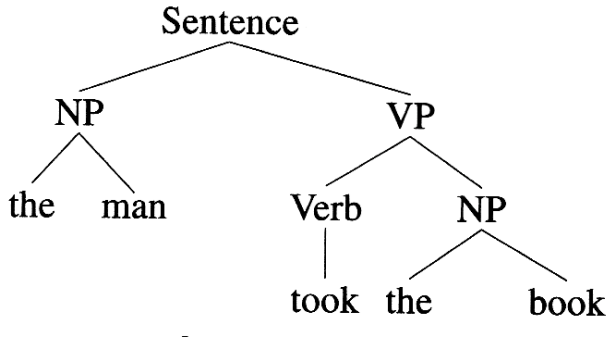
- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book



- *this grammar can generate more than one sentence*
- **examples**
  - the man took the book
  - #the book took the man # = semantically odd
  - *other sentences?*
- **add new rule**
  - Verb → bought
- **examples**
  - the man took the book
  - the man bought the book
  - #the book took the man # = semantically odd
  - #the book bought the man

What is a formal grammar?

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book



- **formally**: *a grammar contains the following 4 things*
  - <N,T,P,S>
    - a set of non-terminal symbols (N)
    - a set of terminal symbols (T)
    - production rules (P) of the form
    - a designated start symbol (S)
- **example**
  - Non-terminals: {Sentence,VP,NP,Verb}
  - Terminals: {the,man,book,took}
  - Start symbol: Sentence
  - Production rules: *set of LHS → RHS rules*

Grammar Rules

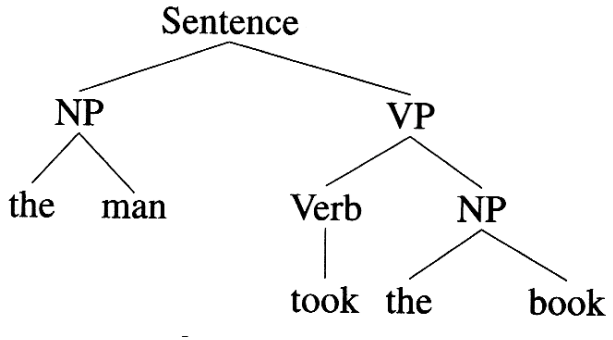
- **Good news!**
  - Prolog supports grammar rules
  - it knows how to interpret them (directly)
  - it can use grammar rules supplied by the user to construct a derivation automatically

Prolog Grammar Rules

- Prolog's version of grammar rules:
  - Definite Clause Grammar (DCG)
- **Prolog's format**
  - terminals and non-terminal symbols begin with lowercase letters
    - e.g. `sentence`, `vp`, `np`, `book`, `took`
    - *Note: variables begin with an uppercase letter (or underscore)*
  - `-->`
    - is the rewrite symbol
  - terminals are enclosed in square brackets to distinguish them from non-terminals (*list notation*)
    - e.g. `[the], [book], [took]`
  - comma (`,`) is the concatenation symbol
  - semicolon (`;`) is the disjunction symbol
  - a period (`.`) is required at the end of all DCG rules

Prolog Grammar Rules

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book



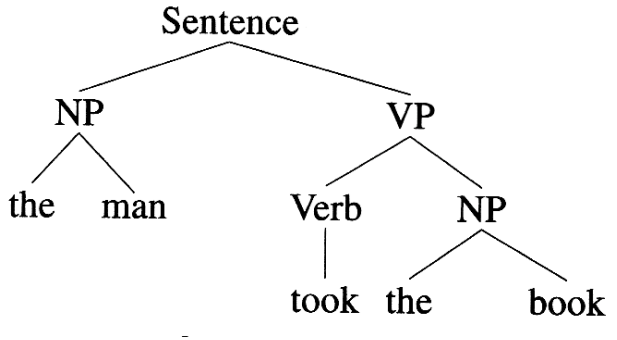
- **Prolog DCG version**

```
sentence --> np, vp.
vp --> verb, np.
verb --> [took].
np --> [the], [man].
np --> [the], [book].
```

- **Important Note**
  - don't enter these rules into the database using assert/1.
  - **Use a file**.

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> [the], [man].`
  - `np --> [the], [book].`



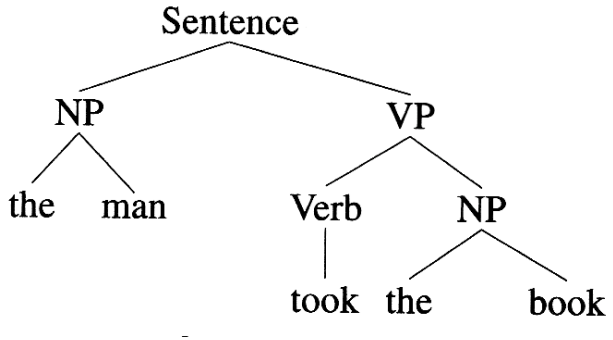
query:

- `?- sentence(S,[]).`
- `S` = sentence (as a list)
- `[]` = empty list

- i.e. *call the start symbol as a predicate and*
- *supply two arguments, a list and an empty list*

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> [the], [man].`
  - `np --> [the], [book].`



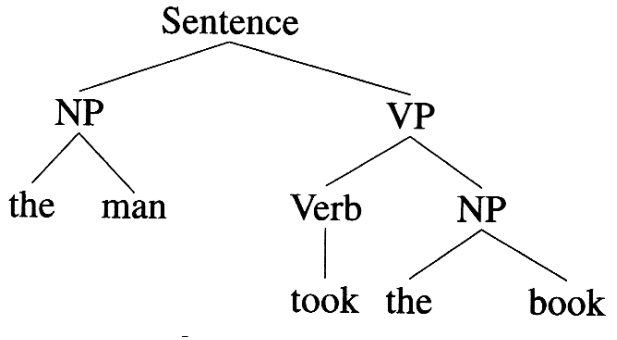
example queries

- `?- sentence([the,man,took,the,book],[]).`
- `Yes`
- "the man took the book" ***is a member of the language generated by the grammar***

- `?- sentence([man,took,the,book],[]).`
- `No`
- "man took the book" ***is not in*** the grammar
- "man took the book" ***is not generated*** by the grammar

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> [the], [man].`
  - `np --> [the], [book].`



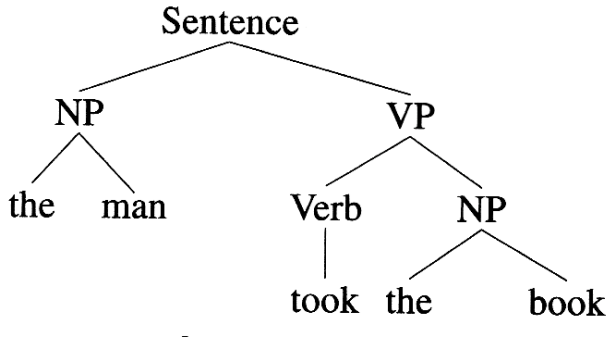
other queries

- `?- sentence([the,man,took,X,book],[]).`
- `X = the`

- `?- sentence(S,[]).`
- `S = [the, man, took, the, man] ;`
- `S = [the, man, took, the, book] ;`
- `S = [the, book, took, the, man] ;`
- `S = [the, book, took, the, book] ;`
- `No`

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> [the], [man].`
  - `np --> [the], [book].`



notes

- `np --> [the,man].` **OK**
- `np --> [the,book].` **OK**

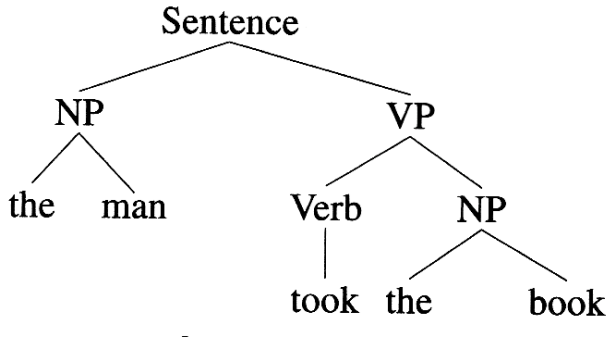
more grammar

det = determiner

- `np --> det, [man].`
- `np --> det, [book].`
- `det --> [the].`
- `det --> [a].`

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> det, [man].`
  - `np --> det, [book].`
  - `det --> [the].`
  - `det --> [a].`



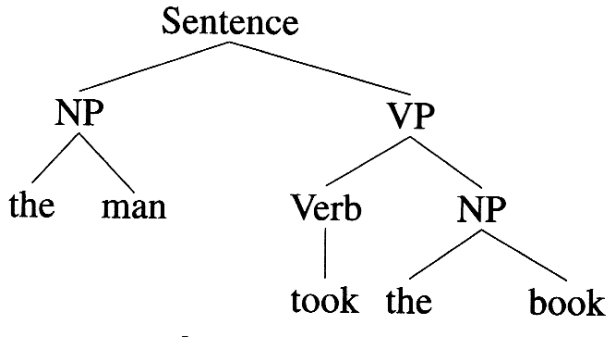
query

- `?- sentence(S,[]).`
- generates 16 different answers for S

- 2 choices for det — a, the
- 2 choices for head noun — man, book
- total of 4 different choices for NP — `(a|(the))((man)|(book))`
- 2 choices for NP — as subject, as object
- total = $4^2 = 16$

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> det, [man].`
  - `np --> det, [book].`
  - `det --> [the].`
  - `det --> [a].`



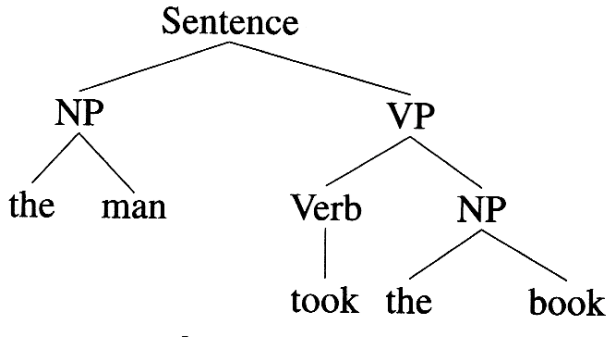
query

- `?- sentence([the,man,took|L],[]).`
- `L = [the, man] ;`
- `L = [a, man] ;`
- `L = [the, book] ;`
- `L = [a, book] ;`
- `No`

4 choices

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> det, [man].`
  - `np --> det, [book].`
  - `det --> [the].`
  - `det --> [a].`



query

- `?- sentence([X,man,took,X,book],[]).`
- `X = the ;`
- `X = a ;`
- `No`

2 choices

Putting a Meaning Together from Pieces

- Chapter ties into what we've been doing:
  - driven by syntax
  - we're going to compute meaning in parts

Putting a Meaning Together from Pieces

- 2.2 Incomplete Propositions

- Shelby barks barks(shelby).

- barks ???

Putting a Meaning Together from Pieces

- 2.2 Incomplete Propositions

- Shelby barks barks(shelby).

- barks barks(X).
  - predicate
  - = unsaturated proposition

Putting a Meaning Together from Pieces

- 2.3 Saturation
  - Shelby barks — barks(shelby).
  - barks — barks(X).
  - Shelby — shelby
- **predication**:
  - relation between predicate barks(X) and its subject shelby
  - *barks* is "***predicated of***" *shelby*
  - i.e. barks(X) and X = *shelby*

Putting a Meaning Together from Pieces

- 2.4 Compositionality
  - (discrete) infinity and creativity of language (new phrases)
  - **Principle of Compositionality**
    - Meaning(Phrase) =
      composition of
      Meaning($SubPart_1$),
      Meaning($SubPart_2$)
      and so on...
  - Example: Shelby barks

Putting a Meaning Together from Pieces

- 2.5 Syntax and Semantics
  - Principle of Compositionality can be realized in different ways
  - **Theories of Meaning:**
    - rule-by-rule theories
    - interpretive theories
  - Example:
    - What did John sit on?
    - John sat on what (+ Wh-phrase movement)

A different kind of example

- Think about the meaning of *any* in:
  1. any dog can do that trick
  2. I didn't see any dog
  3. *I saw any dog

  how many meanings does *any* have?

  do you see any potential problems for rule-by-rule theories?

A different kind of example

- Think about the meaning of *any* in:
  1. any dog can do that trick
  2. I didn't see any dog
  3. *I saw any dog

  how many meanings does *any* have?
  a) "free choice" *any*
  b) negative polarity item (NPI) *any*

Quiz

- [5pts]
- give meaning fragments for:
  - John
  - likes Mary
  - likes
  - in “John likes Mary” corresponds to likes(john,mary).
- give syntactic structures for:
  - who is a student and not a baseball fan?
  - who is not a baseball fan or a student?