LING/C SC 581:

Advanced Computational Linguistics

Lecture 4

Today's Topic

- Homework 2 Review
- We must **refamiliarize** ourselves with Prolog
- Definite Clause Grammars (DCG) quick review
  - *introduced last semester*
  - 538 lectures 24–27 (*see course website*)
- Where we are going – context-sensitive grammars (CSG's)

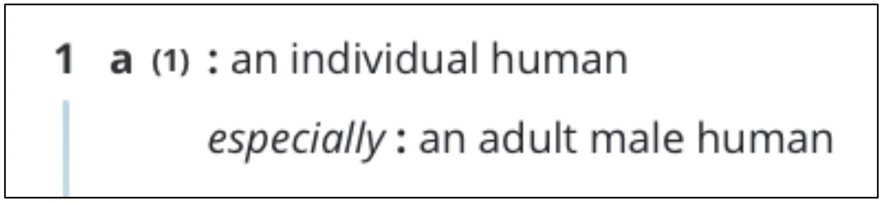
Homework 2 review

- **Garden path sentence**:
  - *humans initially mis-parse the sentence but eventually find the intended reading*.
- Q1: Explain why this is a garden path sentence:
  - *The old man the boat*

  Which word causes the garden path?

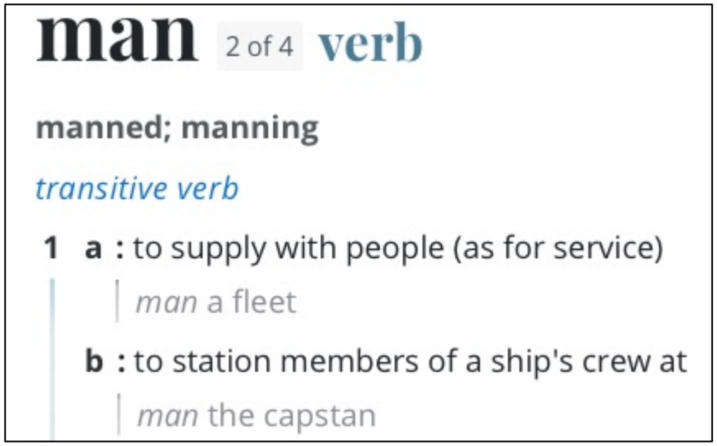
Homework 2 review

- Which word causes the garden path?
  - *The old man the boat*
- Merriam Webster online:
  - https://www.merriam-webster.com/dictionary/man



1 **a** (1) : an individual human

especially : an adult male human



man 2 of 4 **verb**

manned; manning

transitive verb

1 **a** : to supply with people (as for service)

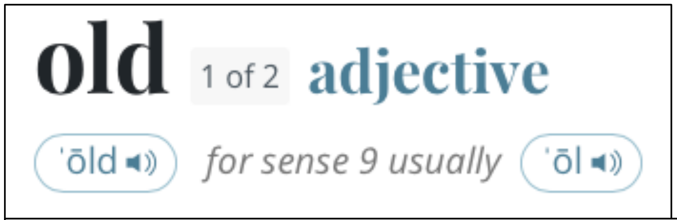
man a fleet

b : to station members of a ship's crew at

man the capstan

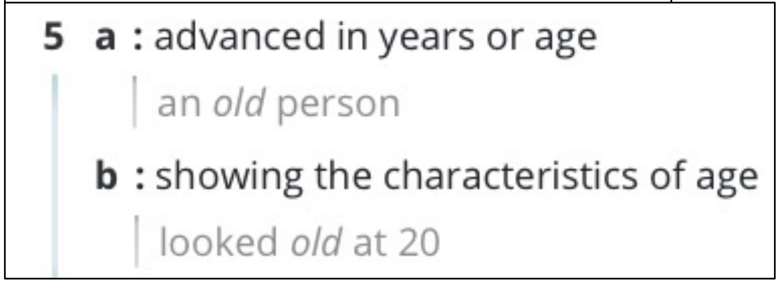
Homework 2 review

- Which word causes the garden path?
  - *The old man the boat*
- Merriam Webster online:
  - https://www.merriam-webster.com/dictionary/man



old 1 of 2 **adjective**

ˈōld for sense 9 usually ˈōl

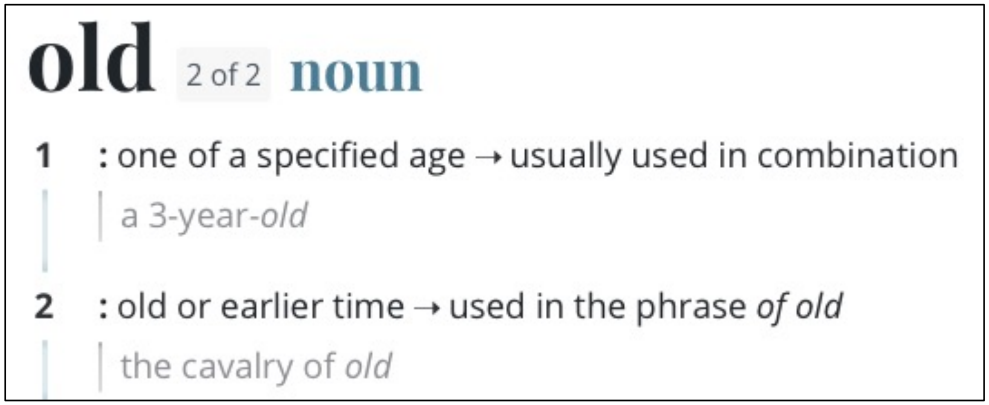


5 **a** : advanced in years or age

an *old* person

b : showing the characteristics of age

looked *old* at 20



old 2 of 2 **noun**

1 : one of a specified age → usually used in combination

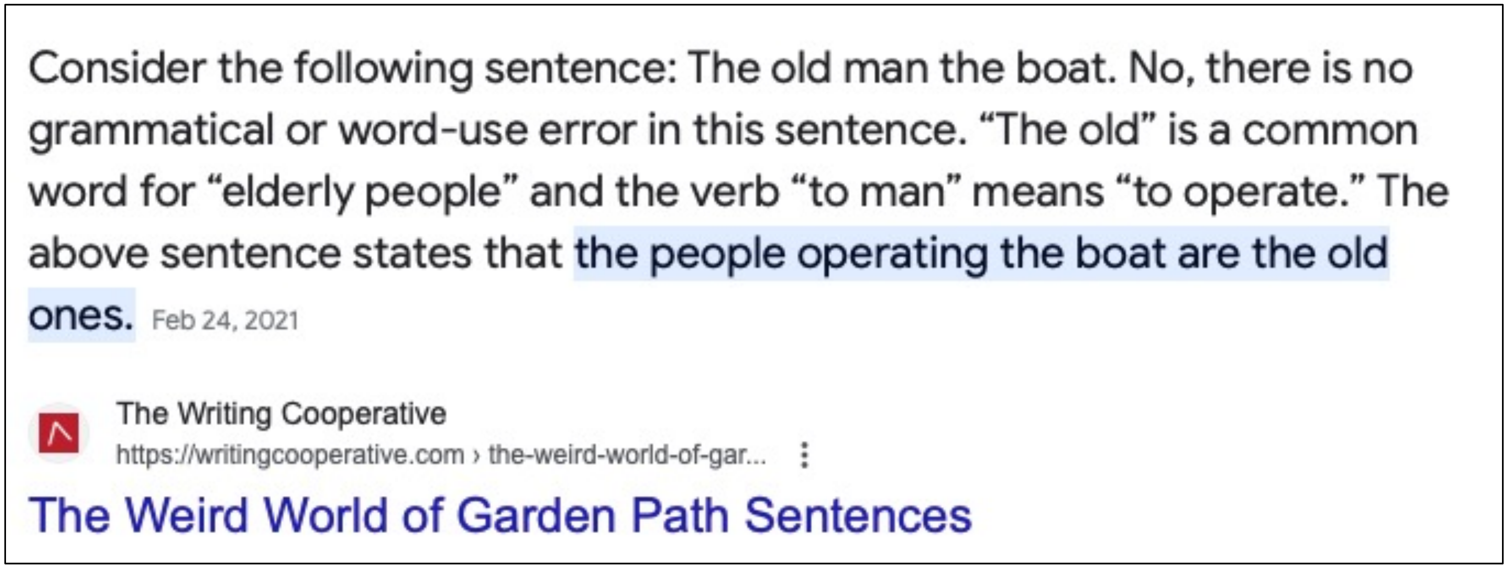
a 3-year-*old*

2 : old or earlier time → used in the phrase *of old*

the cavalry of *old*

Homework 2 review

- many explanations exist on the internet!
- ChatGPT knows about garden path sentences



Consider the following sentence: The old man the boat. No, there is no grammatical or word-use error in this sentence. “The old” is a common word for “elderly people” and the verb “to man” means “to operate.” The above sentence states that the people operating the boat are the old ones. Feb 24, 2021

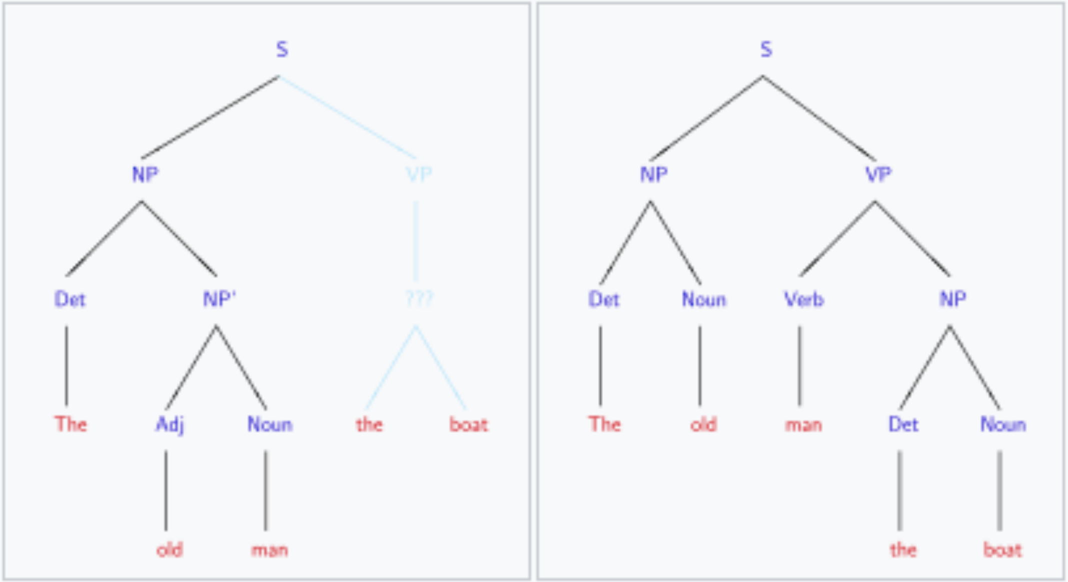
The Writing Cooperative

https://writingcooperative.com › the-weird-world-of-gar...

The Weird World of Garden Path Sentences

Homework 2 review

https://en.wikipedia.org/wiki/Garden-path_sentence



Suggesting uncompletable parse

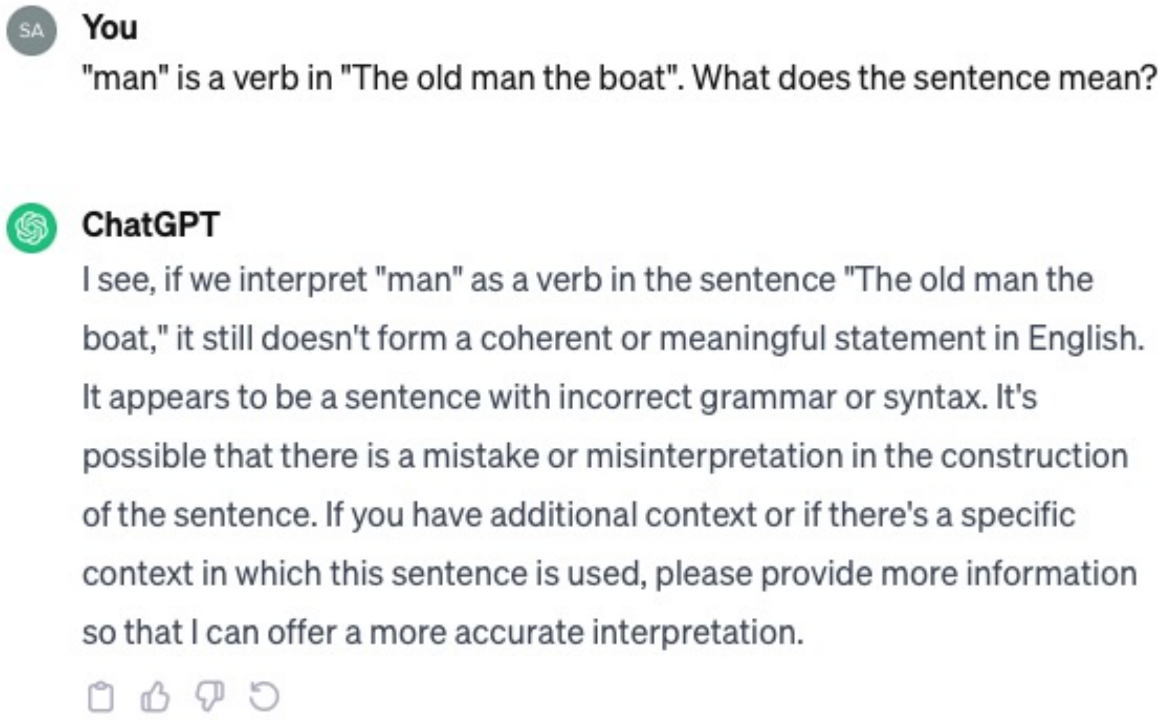
Less obvious complete parse

Why do we garden path?

Possible explanation:

- frequency of sequence DET ADJ NOUN blocks possibility that NOUN is also a VERB

Homework 2



You

"man" is a verb in "The old man the boat". What does the sentence mean?

ChatGPT

I see, if we interpret "man" as a verb in the sentence "The old man the boat," it still doesn't form a coherent or meaningful statement in English. It appears to be a sentence with incorrect grammar or syntax. It's possible that there is a mistake or misinterpretation in the construction of the sentence. If you have additional context or if there's a specific context in which this sentence is used, please provide more information so that I can offer a more accurate interpretation.

Convincing ChatGPT it is wrong

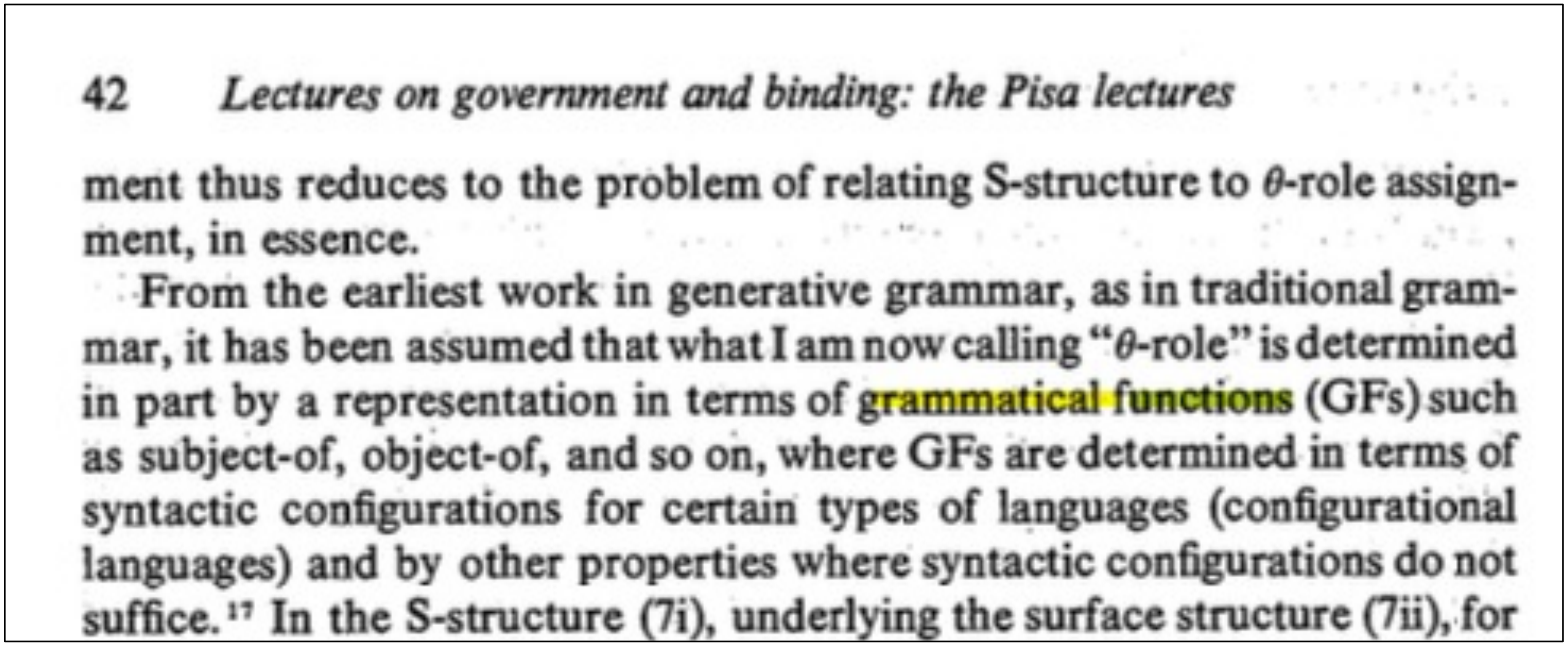
- Involves Knowledge of Language we take for granted:
  1. verbs vs. nouns
  2. word senses
  3. notion of a grammatical relation
  4. notion of a grammatical function

what are these precisely?

does ChatGPT understand them?

Chomsky *LGB* (1980)

- GFs: subject-of, object-of



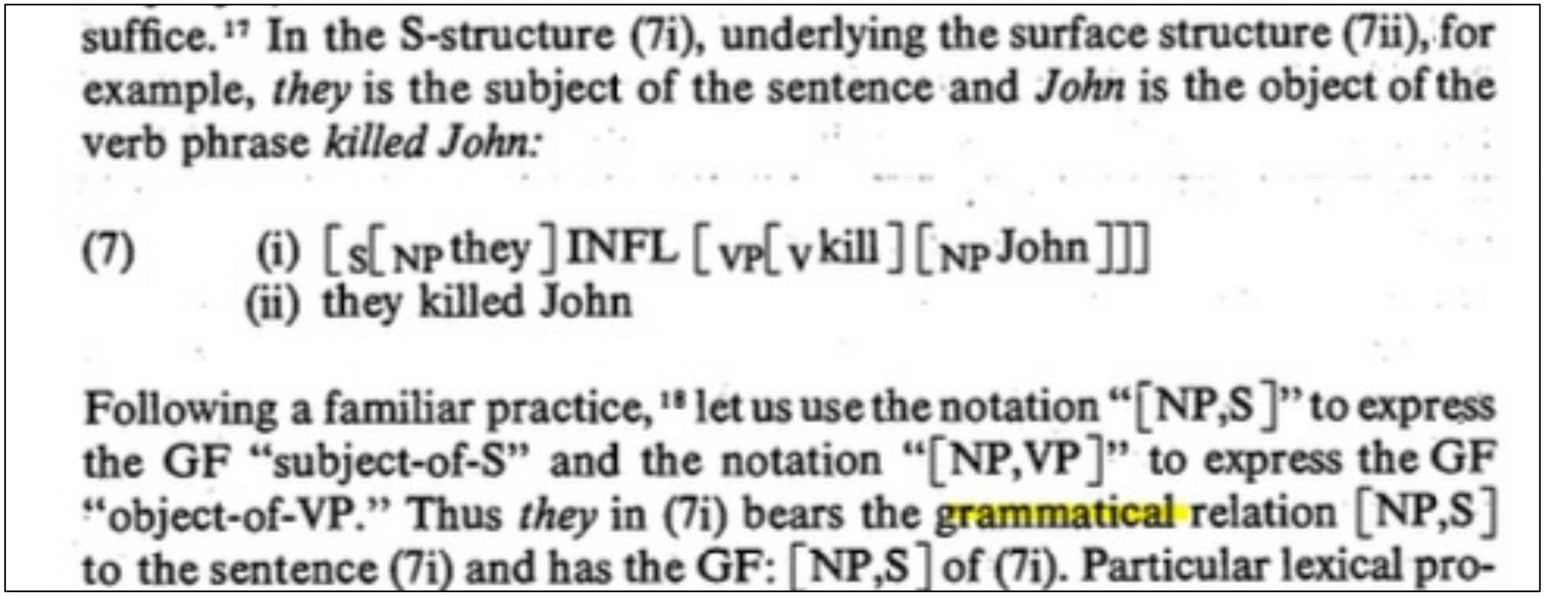
42 *Lectures on government and binding: the Pisa lectures*

ment thus reduces to the problem of relating S-structure to θ-role assignment, in essence.

From the earliest work in generative grammar, as in traditional grammar, it has been assumed that what I am now calling "θ-role" is determined in part by a representation in terms of grammatical functions (GFs) such as subject-of, object-of, and so on, where GFs are determined in terms of syntactic configurations for certain types of languages (configurational languages) and by other properties where syntactic configurations do not suffice.[17] In the S-structure (7i), underlying the surface structure (7ii), for

Chomsky *LGB* (1980)

- configurationally defined [$_{S}$ NP …], [$_{VP}$ NP …]

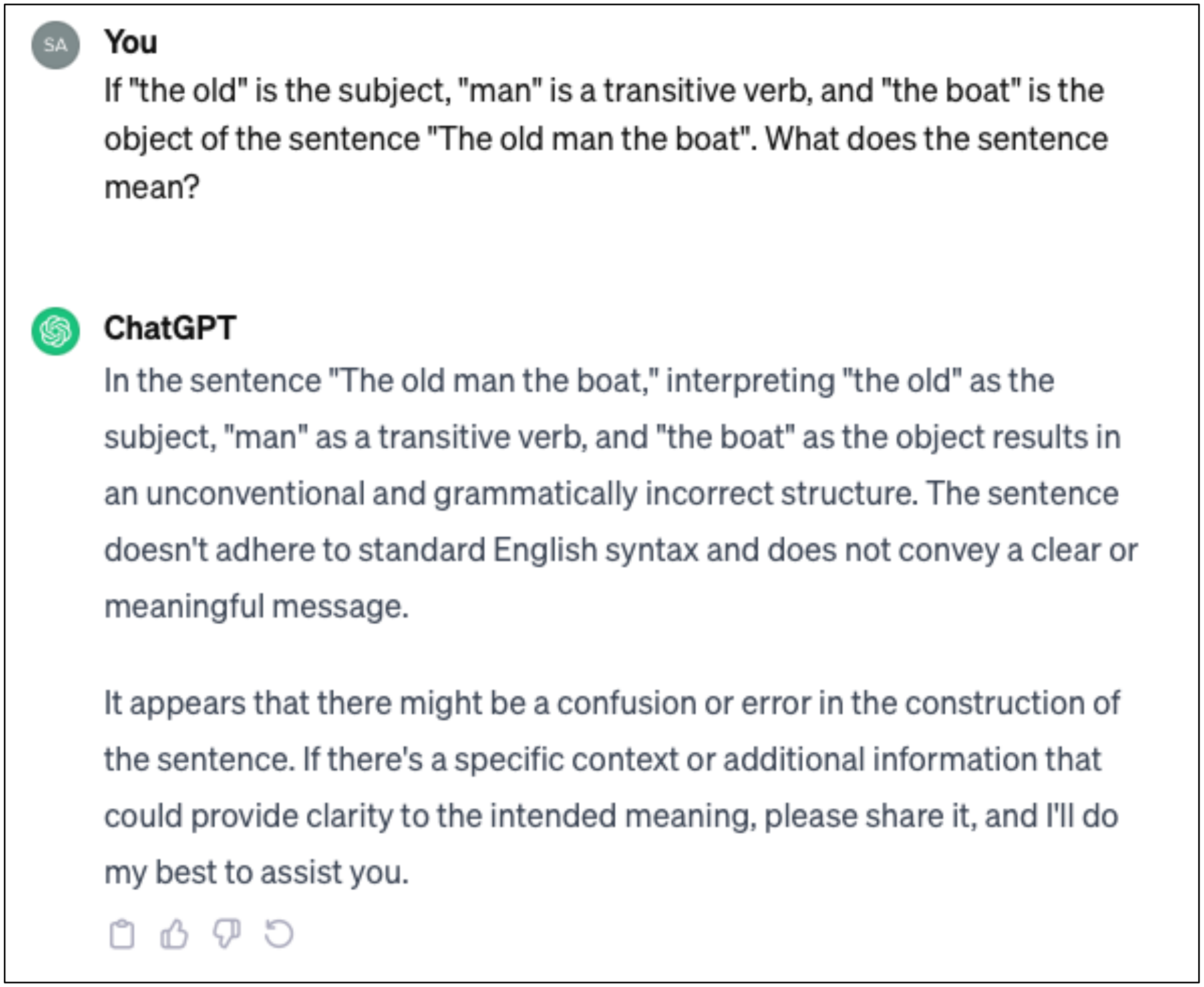


suffice.[17] In the S-structure (7i), underlying the surface structure (7ii), for example, *they* is the subject of the sentence and *John* is the object of the verb phrase *killed John:*

(7) (i) [$_{S}$[$_{NP}$ they] INFL [$_{VP}$[$_{V}$ kill] [$_{NP}$ John]]]
(ii) they killed John

Following a familiar practice,[18] let us use the notation "[NP,S]" to express the GF "subject-of-S" and the notation "[NP,VP]" to express the GF "object-of-VP." Thus *they* in (7i) bears the grammatical relation [NP,S] to the sentence (7i) and has the GF: [NP,S] of (7i). Particular lexical pro-

Homework 2



You

If "the old" is the subject, "man" is a transitive verb, and "the boat" is the object of the sentence "The old man the boat". What does the sentence mean?

ChatGPT

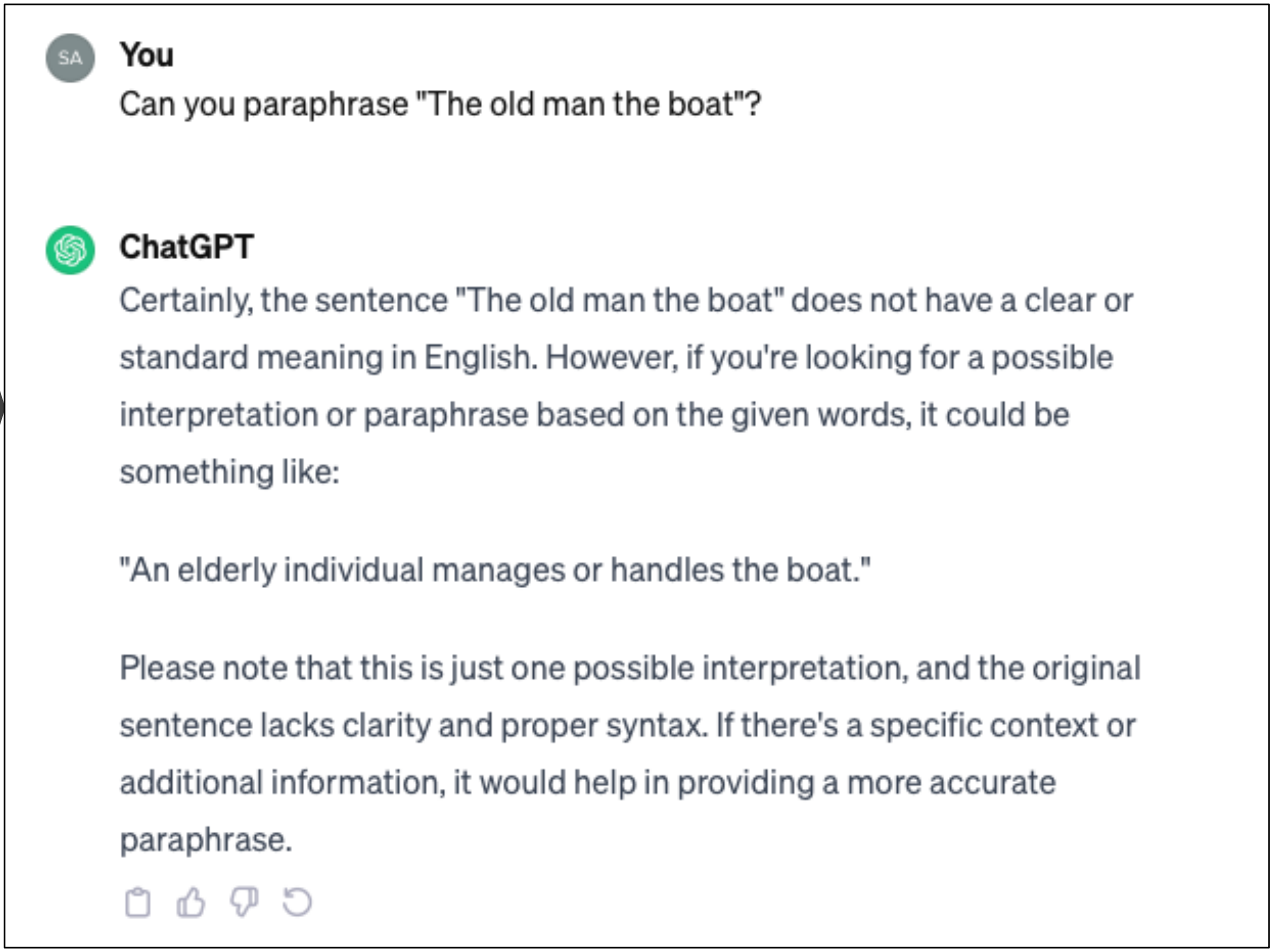
In the sentence "The old man the boat," interpreting "the old" as the subject, "man" as a transitive verb, and "the boat" as the object results in an unconventional and grammatically incorrect structure. The sentence doesn't adhere to standard English syntax and does not convey a clear or meaningful message.

It appears that there might be a confusion or error in the construction of the sentence. If there's a specific context or additional information that could provide clarity to the intended meaning, please share it, and I'll do my best to assist you.

Convincing ChatGPT it is wrong

- More informally, some of you used terms like:
  - hidden verb
  - culprit
  - etc.

Homework 2



You

Can you paraphrase "The old man the boat"?

ChatGPT

Certainly, the sentence "The old man the boat" does not have a clear or standard meaning in English. However, if you're looking for a possible interpretation or paraphrase based on the given words, it could be something like:

"An elderly individual manages or handles the boat."

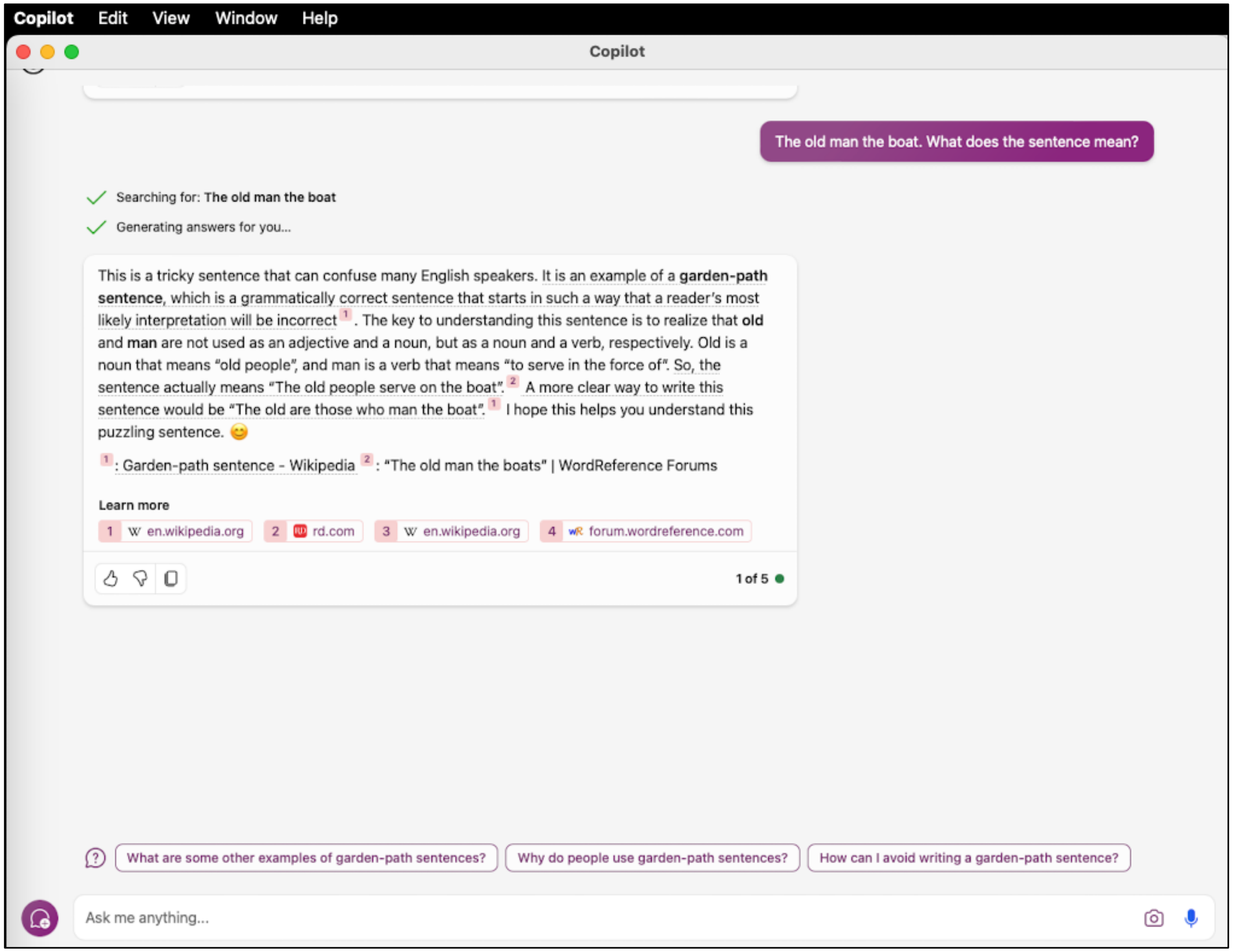
Please note that this is just one possible interpretation, and the original sentence lacks clarity and proper syntax. If there's a specific context or additional information, it would help in providing a more accurate paraphrase.

Microsoft Copilot

Your everyday AI companion

Designed for iPad. Not verified for macOS.

Microsoft CoPilot (*uses* ChatGPT 4)



The old man the boat. What does the sentence mean?

✓ Searching for: **The old man the boat**

✓ Generating answers for you…

This is a tricky sentence that can confuse many English speakers. It is an example of a **garden-path sentence**, which is a grammatically correct sentence that starts in such a way that a reader's most likely interpretation will be incorrect [1]. The key to understanding this sentence is to realize that **old** and **man** are not used as an adjective and a noun, but as a noun and a verb, respectively. Old is a noun that means "old people", and man is a verb that means "to serve in the force of". So, the sentence actually means "The old people serve on the boat". [2] A more clear way to write this sentence would be "The old are those who man the boat". [1] I hope this helps you understand this puzzling sentence. 😊

[1]: Garden-path sentence - Wikipedia [2]: "The old man the boats" | WordReference Forums

Learn more

1 en.wikipedia.org 2 rd.com 3 en.wikipedia.org 4 forum.wordreference.com

1 of 5
What are some other examples of garden-path sentences?
Why do people use garden-path sentences?
How can I avoid writing a garden-path sentence?
Ask me anything...

SWI Prolog Cheatsheet

- **At the prompt** `?-`
  1. `halt.` `^D`
  2. `listing.` `listing(`*name*`).`
  3. `[`*filename*`].` loads *filename.pl*
  4. `trace.`
  5. `notrace.`
  6. `debug.`
  7. `nodebug.`
  8. `spy(`*name*`).`
  9. `pwd.`
  10. `working_directory(_,Y).`
      switch directories to Y
- Anytime
  - `^C` (then **a**(bort) or **h**(elp) for other options)

Notation:

`\+`	negation
`,`	conjunction
`;`	disjunction
`:-`	if

Facts:

`predicate(`*Args*`).`

Rules:

`p(`*Args*`) :- q(`*Args*`) ,.., r(`*Args*`).`

Data structures:

list: `[a,..b]`

empty list: `[]`

head/tail: `[`*head*`|`*List*`]`

Atom:

name, number

Term:

`functor(arguments)`

`arguments:` comma-separated terms/atoms

Derivations

- Prolog's computation rule:
  - Try first matching (**grammar**) rule in the database
    (but remember other possibilities for backtracking)
  - Backtrack if matching rule leads to failure (or if asked by the user typing `;`)
  - Auto-management of alternative possibilites:
    - undo variable bindings (i.e. *undo assignments*) and try next matching rule
- For grammars:
  - Top-down left-to-right derivations
    - **left-to-right** = expand leftmost nonterminal first
    - Leftmost expansion done recursively = **depth-first**

Definite Clause Grammars (DCG)

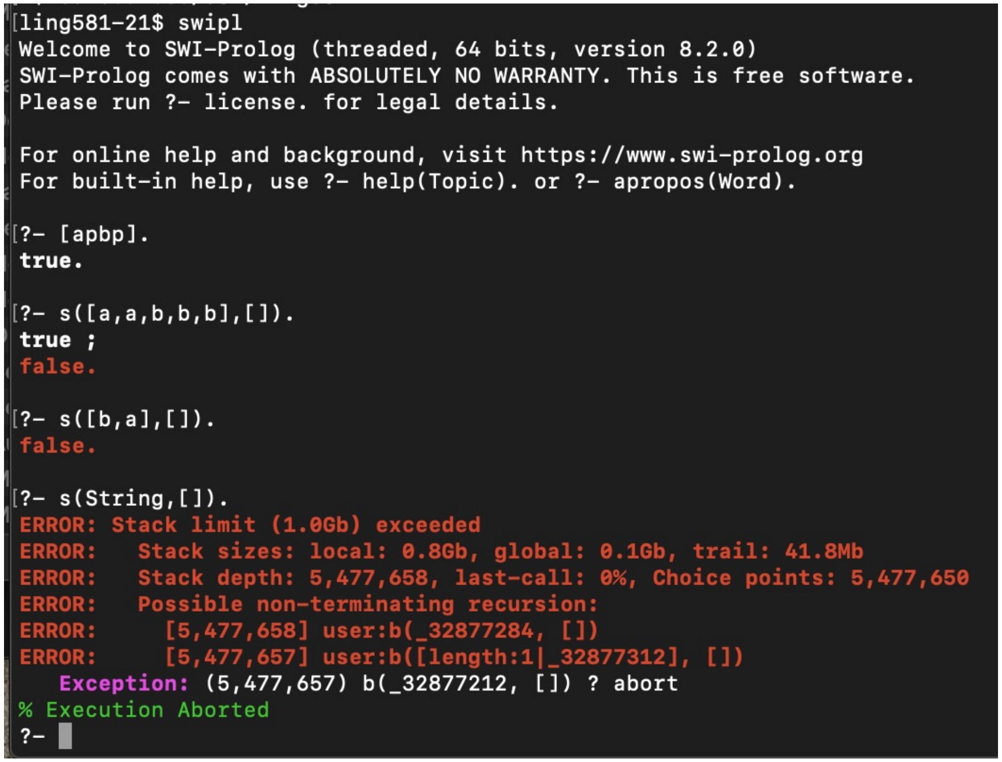
- a grammar is code, could be a recognizer program:
  - *no need to write a separate grammar rule interpreter* (in this case)
- **Example query**
  - `?- s([a,a,b,b,b],[]).` Yes
- **Note**:
  - Syntax of DCGs:
    - ---> "expands to"
    - terminal symbol enclosed in square brackets: [*terminal*]
    - non-terminal symbol, otherwise
  - Query uses the start symbol `s` with two arguments:
  - (1) sequence (as a list) to be recognized and
  - (2) the empty list `[]`

Grammar for a^+b^+

apbp.prolog:

```
s --> [a],b.
b --> [a],b.
b --> [b],c.
b --> [b].
c --> [b],c.
c --> [b].
```

Definite Clause Grammars (DCG)

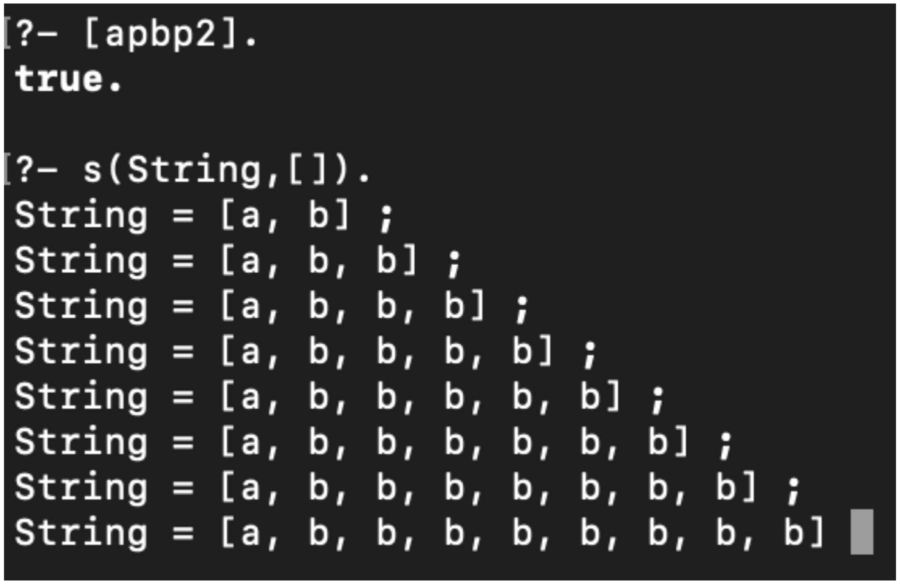


```
[ling581-21$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

[?- [apbp].
true.

[?- s([a,a,b,b,b],[]).
true ;
false.

[?- s([b,a],[]).
false.

[?- s(String,[]).
ERROR: Stack limit (1.0Gb) exceeded
ERROR:   Stack sizes: local: 0.8Gb, global: 0.1Gb, trail: 41.8Mb
ERROR:   Stack depth: 5,477,658, last-call: 0%, Choice points: 5,477,650
ERROR:   Possible non-terminating recursion:
ERROR:     [5,477,658] user:b(_32877284, [])
ERROR:     [5,477,657] user:b([length:1|_32877312], [])
   Exception: (5,477,657) b(_32877212, []) ? abort
% Execution Aborted
?- 
```

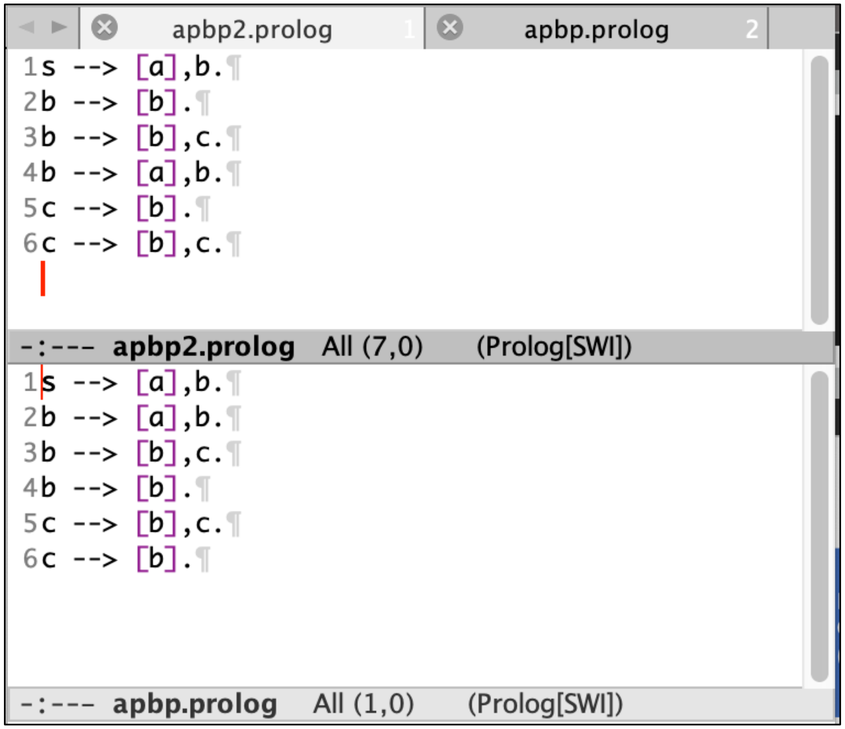
Infinite Loop:
- Doesn't enumerate at all!

Definite Clause Grammars (DCG)

Partial enumerator only! Why?



```
[?- [apbp2].
true.

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


apbp2.prolog
1
apbp.prolog
2

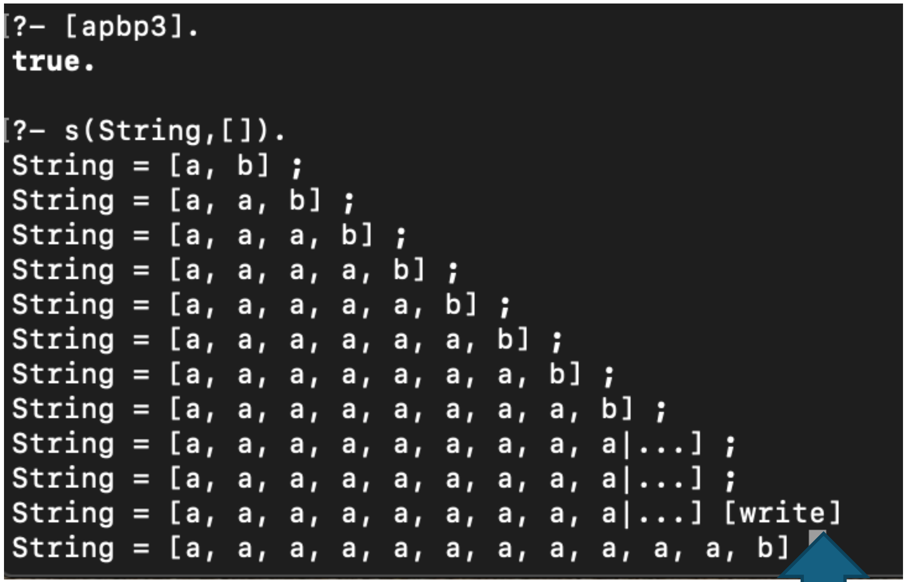
```
s --> [a],b.
b --> [b].
b --> [b],c.
b --> [a],b.
c --> [b].
c --> [b],c.
```

-:--- apbp2.prolog All (7,0) (Prolog[SWI])

```
s --> [a],b.
b --> [a],b.
b --> [b],c.
b --> [b].
c --> [b],c.
c --> [b].
```

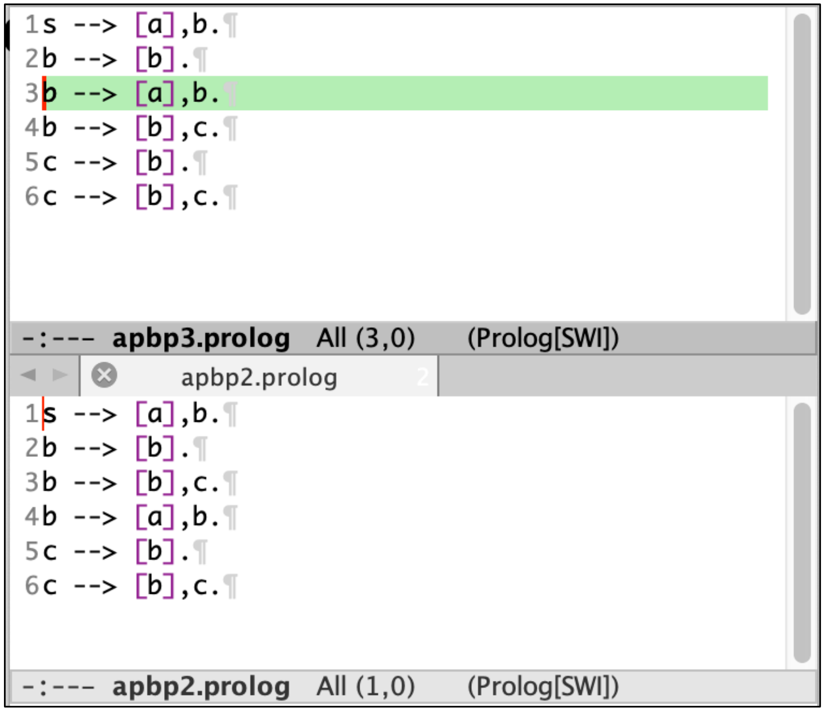
-:--- apbp.prolog All (1,0) (Prolog[SWI])

Definite Clause Grammars (DCG)



```
?- [apbp3].
true.

?- s(String,[]).
String = [a, b] ;
String = [a, a, b] ;
String = [a, a, a, b] ;
String = [a, a, a, a, b] ;
String = [a, a, a, a, a, b] ;
String = [a, a, a, a, a, a, b] ;
String = [a, a, a, a, a, a, a, b] ;
String = [a, a, a, a, a, a, a, a, b] ;
String = [a, a, a, a, a, a, a, a, a|...] ;
String = [a, a, a, a, a, a, a, a, a|...] ;
String = [a, a, a, a, a, a, a, a, a|...] [write]
String = [a, a, a, a, a, a, a, a, a, a, a, b]
```

type w



```
s --> [a],b.
b --> [b].
b --> [a],b.
b --> [b],c.
c --> [b].
c --> [b],c.
```

-:--- apbp3.prolog All (3,0) (Prolog[SWI])
apbp2.prolog

```
s --> [a],b.
b --> [b].
b --> [b],c.
b --> [a],b.
c --> [b].
c --> [b],c.
```

-:--- apbp2.prolog All (1,0) (Prolog[SWI])

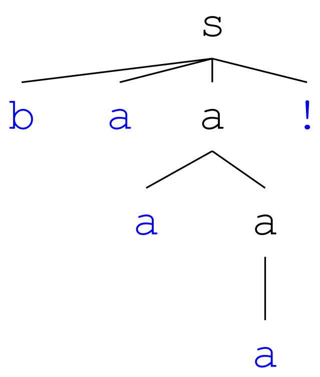
Extra Argument: Parse Tree

- **Recovering a parse tree**
  - when want Prolog to return more than just **true/false** answers
  - in case of **true**, we can compute a syntax tree representation of the parse
  - by adding an extra argument to nonterminals

Example (*sheeptalk.prolog*)

- **DCG (non-regular, context-free):**

```
s --> [b], [a], a, [!].
a --> [a].        (base case)
a --> [a], a.  (recursive case)
```



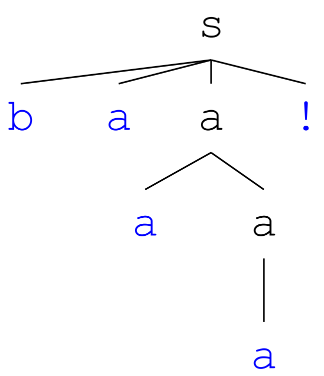
Extra Argument: Parse Tree

- want Prolog to return more than just **true/false** answers

```
$ swipl
Welcome to SWI-Prolog (threaded,
64 bits, version 9.0.4)
?- [sheeptalk].
true.
?- s(String, []).
String = [b, a, a, !] ;
String = [b, a, a, a, !] ;
String = [b, a, a, a, a, !] ;
String = [b, a, a, a, a, a, !] .

?- s([b,a,a,!], []).
true ;
false.

?- s([m,o,o,!], []).
false.
```

Extra Argument: Parse Tree

- **Tree:**

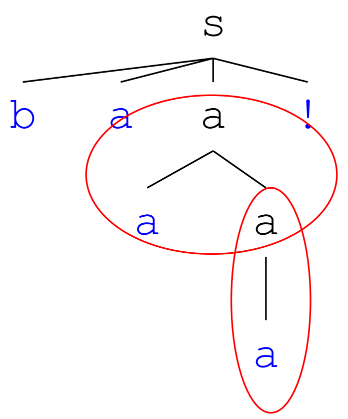


`s(b,a,a(a,a(a)),!)`

- **Prolog term data structure:**
  - hierarchical
  - allows sequencing of arguments
  - $\texttt{functor(arg}_1\texttt{,..,arg}_n\texttt{)}$
  - each $\texttt{arg}_i$ could be another **term** or simple atom

Extra Arguments: Parse Tree

- **Sheeptalk DCG**
  - s --> [b],[a], a, [!].
  - a --> [a]. (base case)
  - a --> [a], a. (right recursive case)
- **base case**
  - a --> [a].
  - a(*subtree*) --> [a].
  - a(a(a)) --> [a].
- **recursive case**
  - a --> [a], a.
  - a(*subtree*) --> [a], a(*subtree*).
  - a(a(a,A)) --> [a], a(A).

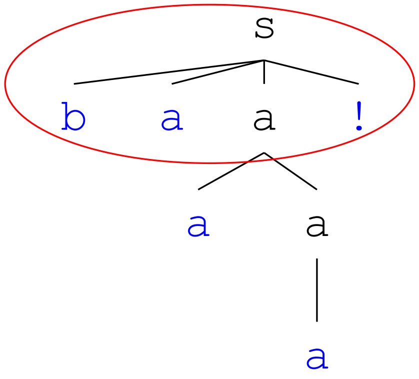


s(b,a,a(a,a(a)),!)

Idea: for each nonterminal, add an argument to store its subtree

Extra Arguments: Parse Tree

- **Prolog grammar**
  - s --> [b], [a], a, [!].
  - a --> [a]. (base case)
  - a --> [a], a. (right recursive case)
- **base and recursive cases**
  - a(a(a)) --> [a].
  - a(a(a,A)) --> [a], a(A).
- **start symbol case**
  - s --> [b], [a], a, [!].
  - s(*tree*) --> [b], [a], a(*subtree*), [!].
  - s(s(b,a,A,!)) --> [b], [a], a(A), [!].



s(b,a,a(a,a(a)),!)

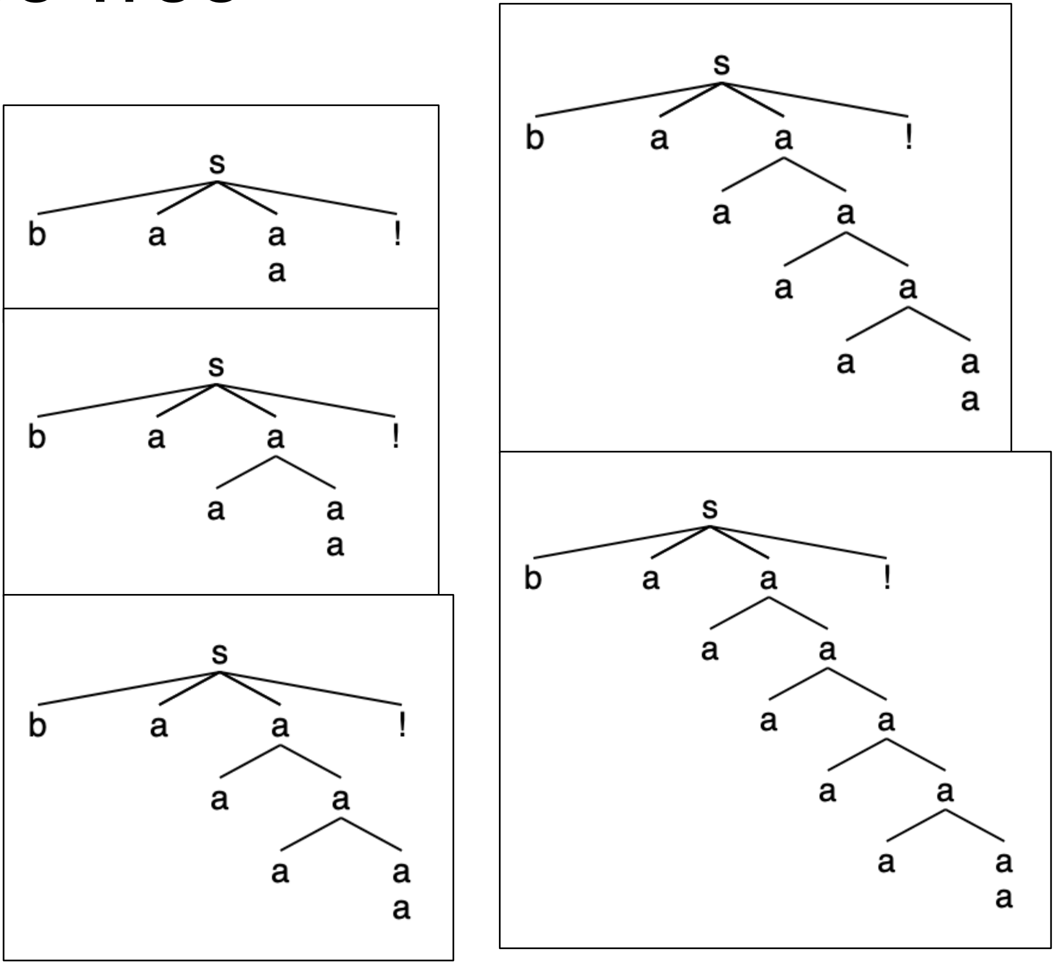
Extra Arguments: Parse Tree

- **Prolog grammar**
  - `sheeptalk.prolog`
  - `s --> [b], [a], a, [!].`
  - `a --> [a].` (base case)
  - `a --> [a], a.` (right recursive case)

- **Equivalent Prolog grammar computing a parse**
  - `sheeptalk2.prolog`
  - `s(s(b,a,A,!)) --> [b], [a], a(A), [!].`
  - `a(a(a)) --> [a].`
  - `a(a(a,A)) --> [a], a(A).`

Extra Arguments: Parse Tree

```
?- [sheeptalk2].
true.
?- s(Parse, String, []).
Parse = s(b, a, a(a), !),
String = [b, a, a, !] ;
Parse = s(b, a, a(a, a(a)), !),
String = [b, a, a, a, !] ;
Parse = s(b, a, a(a, a(a, a(a))), !),
String = [b, a, a, a, a, !] ;
Parse = s(b, a, a(a, a(a, a(a, a(a)))),
!),
String = [b, a, a, a, a, a, !] ;
Parse = s(b, a, a(a, a(a, a(a, a(a,
a(a))))), !),
String = [b, a, a, a, a, a, a, !] .
```



Extra Arguments: Parse Tree

```
?- [sheeptalk2].
true.
?- s(Parse, [b,a,a,!], []).
Parse = s(b, a, a(a), !) ;
false.
?- s(Parse, [m,o,o,!], []).
false.
```

Extra Arguments

- Extra arguments are powerful
  - they allow us to impose (grammatical) constraints and change the expressive power of the system
    - *if used as read-able memory*

- **Example**:
  - $a^n b^n c^n$ n>0 is not a context-free language (type-2, Chomsky hierarchy)
  - *i.e. you cannot write rules of the form* X --> RHS *that generate this language*
  - in fact, it's context-sensitive (type-1, Chomsky hierarchy)