NLP

Introduction to NLP

Semantic Parsing

Semantic Parsing

- Converting natural language to a logical form
  - e.g., executable code for a specific application
- Example:
  - Airline reservations
  - Geographical query systems

Stages of Semantic Parsing

- Input
  - Sentence
- Syntactic Analysis
  - Syntactic structure
- Semantic Analysis
  - Semantic representation

Compositional Semantics

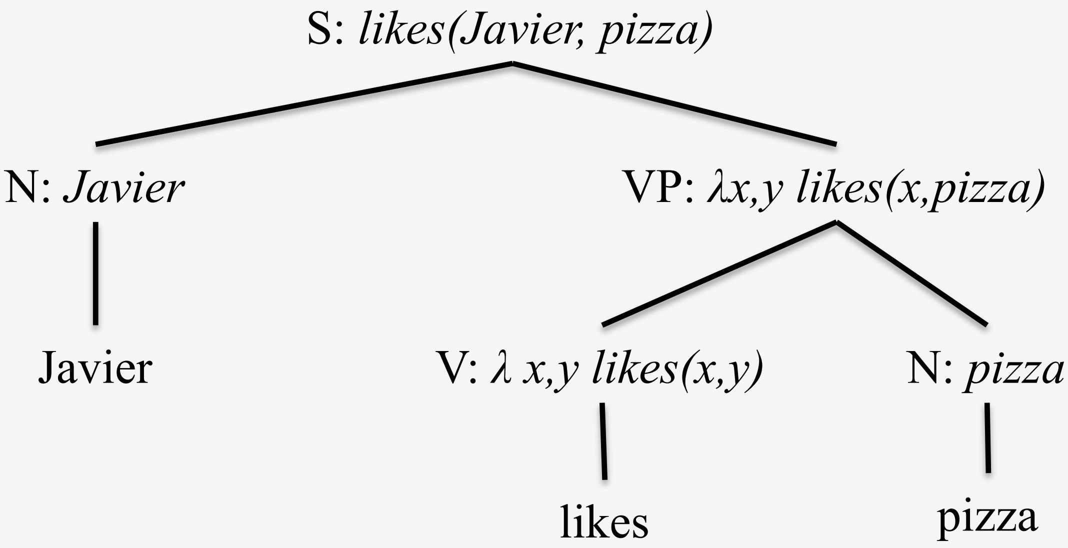
- Add semantic attachments to CFG rules
- Compositional semantics
  - Parse the sentence syntactically
  - Associate some semantics to each word
  - Combine the semantics of words and non-terminals recursively
  - Until the root of the sentence

Example

- Input
  - Javier likes pizza
- Output
  - *like(Javier, pizza)*

Semantic Parsing

- Associate a semantic expression with each node



S: *likes(Javier, pizza)*

N: *Javier*

VP: *λx,y likes(x,pizza)*

Javier

V: *λ x,y likes(x,y)*

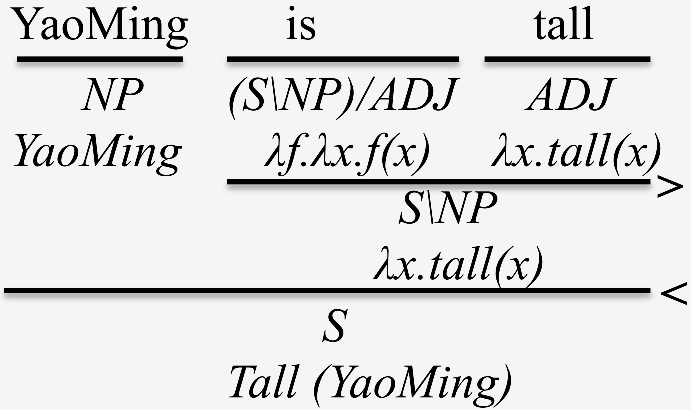
N: *pizza*

likes

pizza

Using CCG (Steedman 1996)

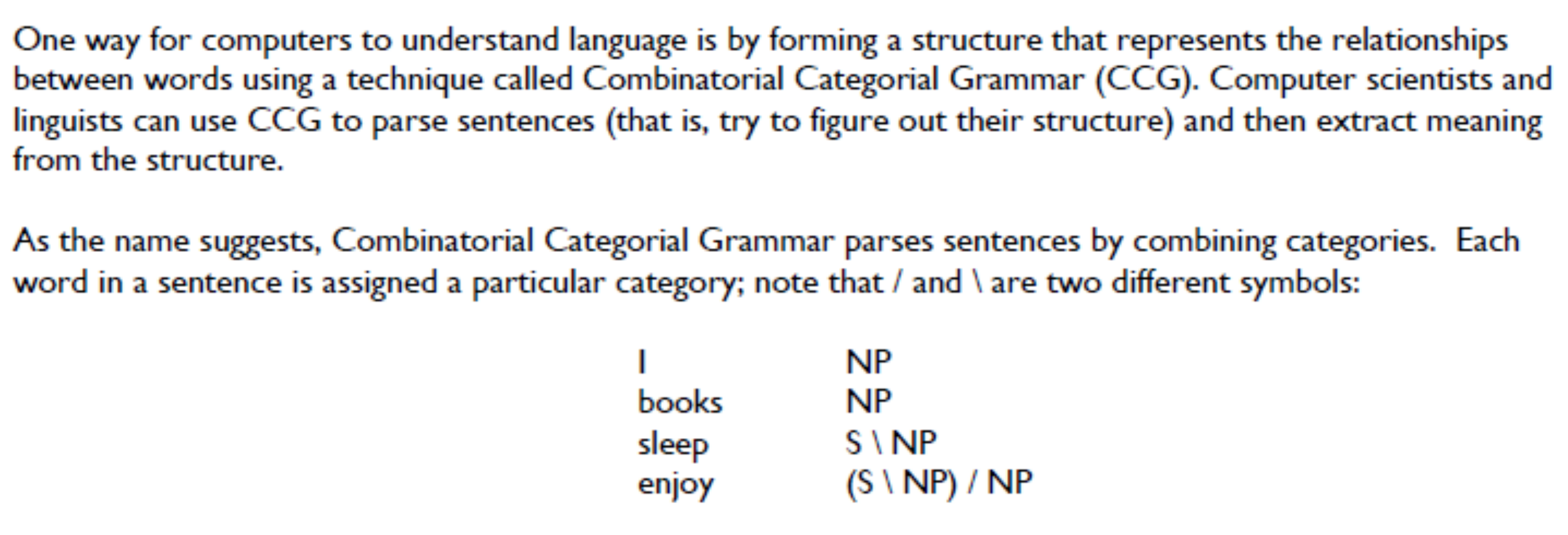
- CCG representations for semantics
  - *ADJ: λx.tall(x)*
  - (S\NP)/ADJ : *λf.λx.f(x)*
  - *NP: YaoMing*



CCG

- NACLO problem from 2014
- Authors: Jonathan Kummerfeld, Aleka Blackwell, and Patrick Littell
- http://www.nacloweb.org/resources/problems/2014/N2014-O.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-OS.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-P.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-PS.pdf

CCG

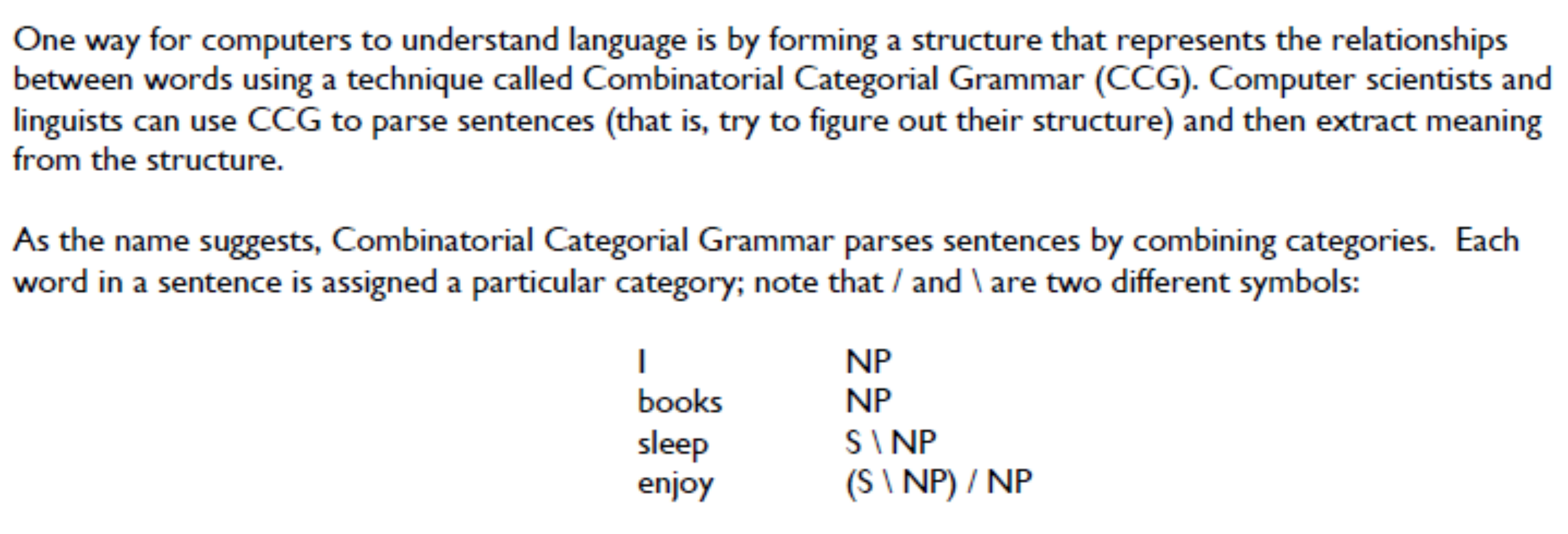


One way for computers to understand language is by forming a structure that represents the relationships between words using a technique called Combinatorial Categorial Grammar (CCG). Computer scientists and linguists can use CCG to parse sentences (that is, try to figure out their structure) and then extract meaning from the structure.

As the name suggests, Combinatorial Categorial Grammar parses sentences by combining categories. Each word in a sentence is assigned a particular category; note that / and \ are two different symbols:

I	NP
books	NP
sleep	S \ NP
enjoy	(S \ NP) / NP

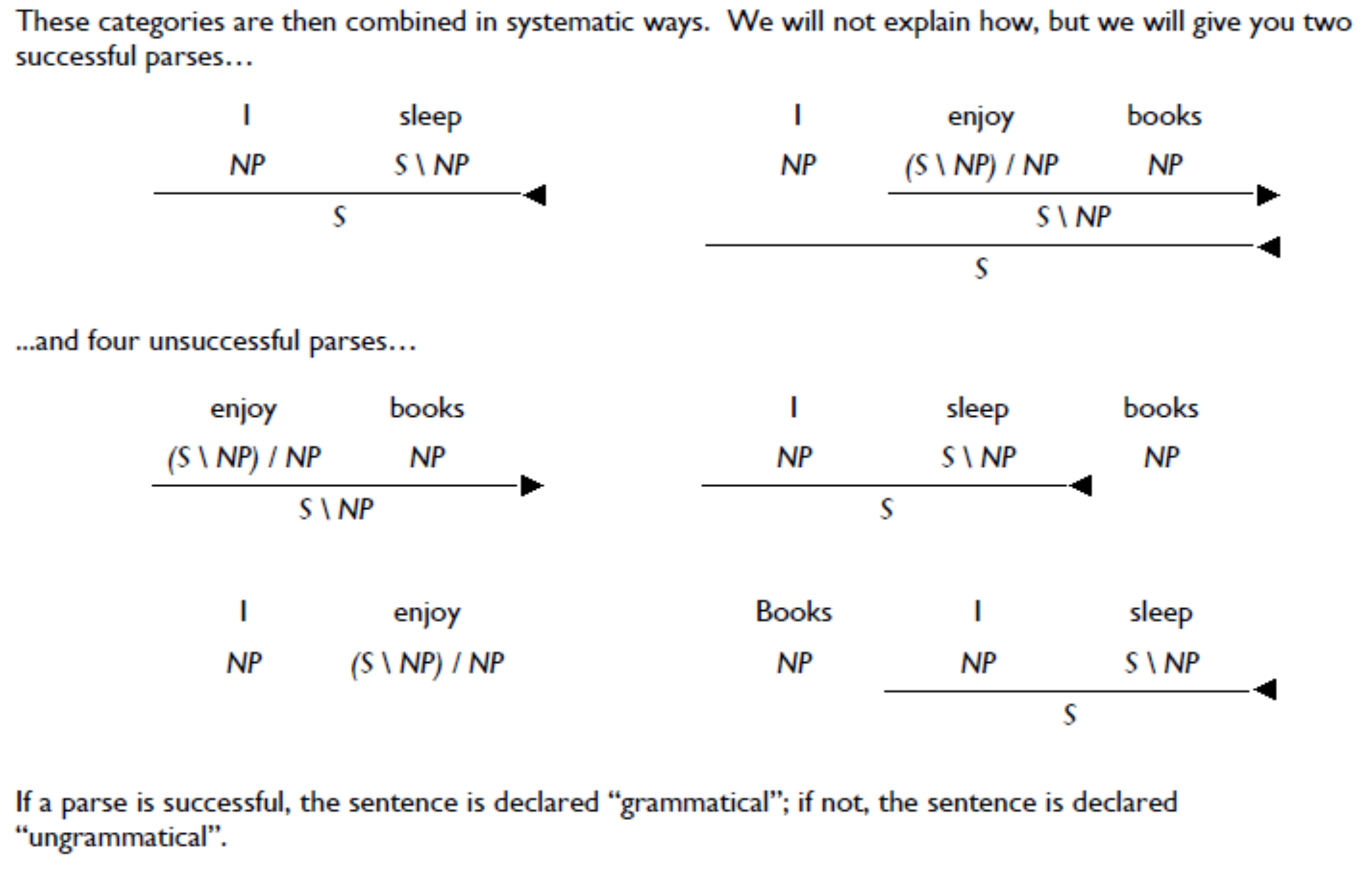
CCG



One way for computers to understand language is by forming a structure that represents the relationships between words using a technique called Combinatorial Categorial Grammar (CCG). Computer scientists and linguists can use CCG to parse sentences (that is, try to figure out their structure) and then extract meaning from the structure.

As the name suggests, Combinatorial Categorial Grammar parses sentences by combining categories. Each word in a sentence is assigned a particular category; note that / and \ are two different symbols:

I	NP
books	NP
sleep	S \ NP
enjoy	(S \ NP) / NP



These categories are then combined in systematic ways. We will not explain how, but we will give you two successful parses…

```
   I          sleep
   NP         S \ NP
   ─────────────────◀
          S
```

```
   I          enjoy           books
   NP         (S \ NP) / NP   NP
              ──────────────────▶
                   S \ NP
   ─────────────────────────────◀
                   S
```

...and four unsuccessful parses…

```
   enjoy           books
   (S \ NP) / NP   NP
   ──────────────────▶
        S \ NP
```

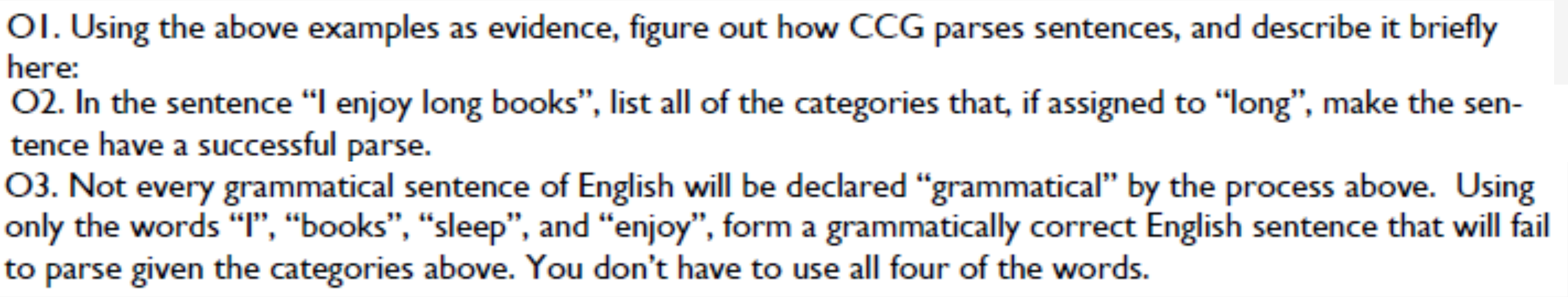
```
   I          sleep       books
   NP         S \ NP      NP
   ─────────────────◀
          S
```

```
   I          enjoy
   NP         (S \ NP) / NP
```

```
   Books      I          sleep
   NP         NP         S \ NP
              ─────────────────◀
                     S
```

If a parse is successful, the sentence is declared “grammatical”; if not, the sentence is declared “ungrammatical”.

CCG

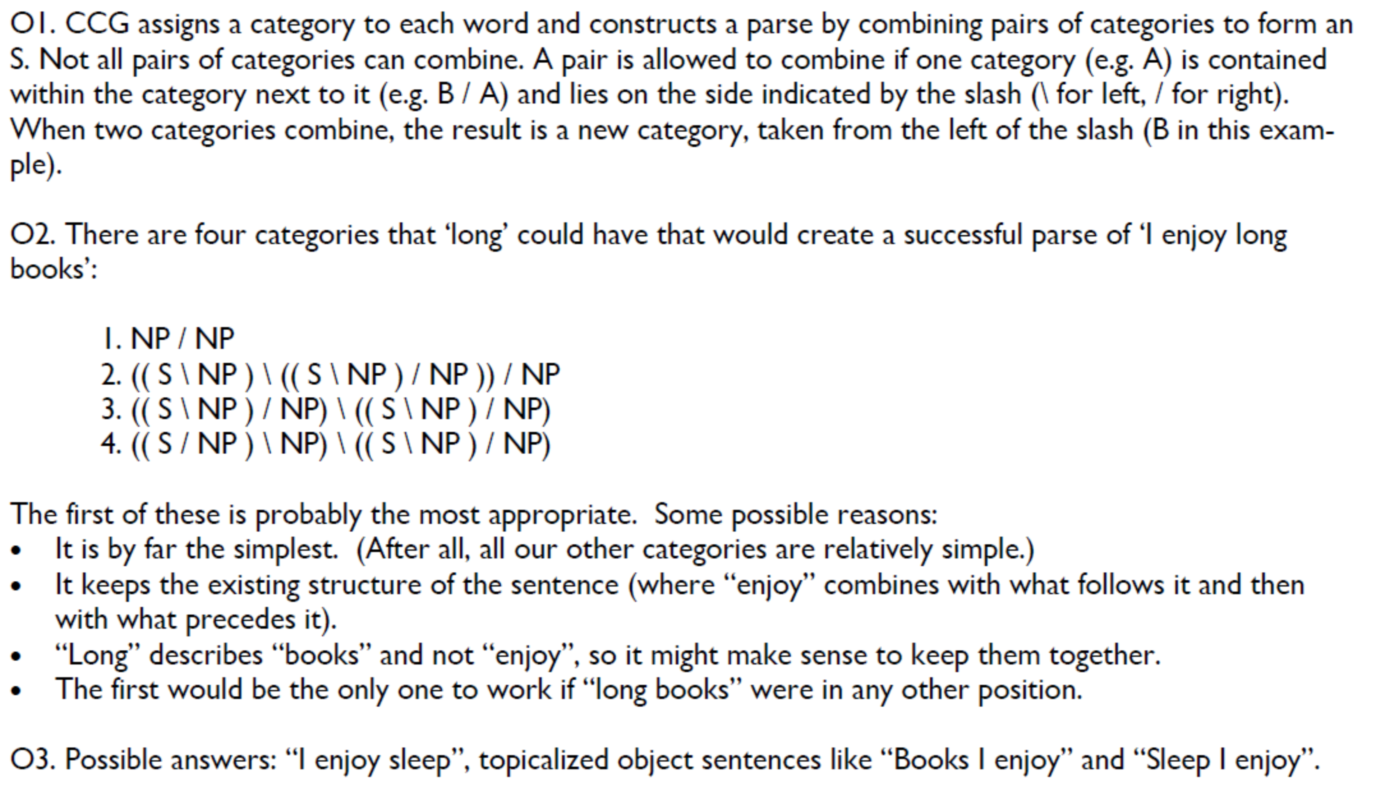


O1. Using the above examples as evidence, figure out how CCG parses sentences, and describe it briefly here:

O2. In the sentence “I enjoy long books”, list all of the categories that, if assigned to “long”, make the sentence have a successful parse.

O3. Not every grammatical sentence of English will be declared “grammatical” by the process above. Using only the words “I”, “books”, “sleep”, and “enjoy”, form a grammatically correct English sentence that will fail to parse given the categories above. You don’t have to use all four of the words.

Answer



O1. CCG assigns a category to each word and constructs a parse by combining pairs of categories to form an S. Not all pairs of categories can combine. A pair is allowed to combine if one category (e.g. A) is contained within the category next to it (e.g. B / A) and lies on the side indicated by the slash (\ for left, / for right). When two categories combine, the result is a new category, taken from the left of the slash (B in this example).

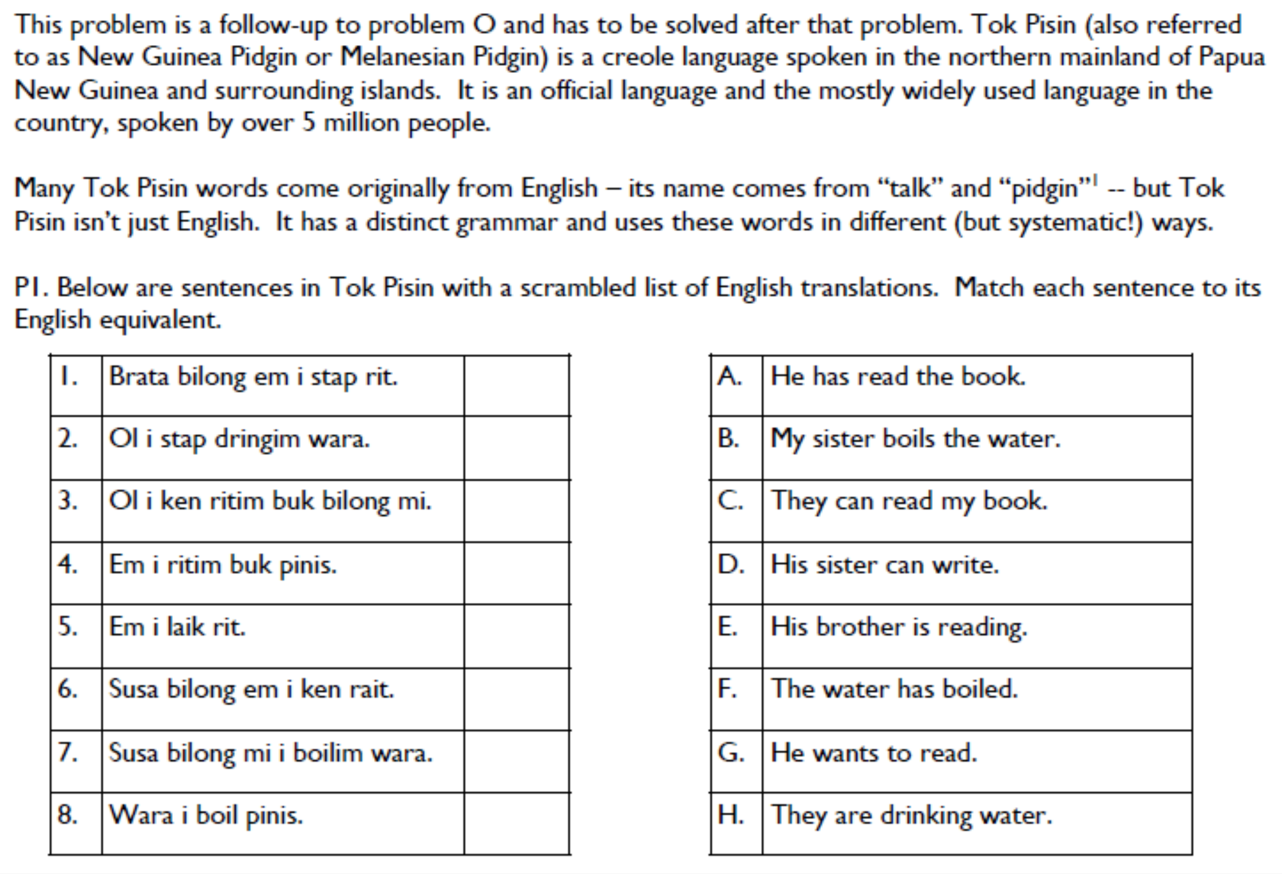
O2. There are four categories that 'long' could have that would create a successful parse of 'I enjoy long books':

1. NP / NP
2. ((S \ NP) \ ((S \ NP) / NP)) / NP
3. ((S \ NP) / NP) \ ((S \ NP) / NP)
4. ((S / NP) \ NP) \ ((S \ NP) / NP)

The first of these is probably the most appropriate. Some possible reasons:

- It is by far the simplest. (After all, all our other categories are relatively simple.)
- It keeps the existing structure of the sentence (where "enjoy" combines with what follows it and then with what precedes it).
- "Long" describes "books" and not "enjoy", so it might make sense to keep them together.
- The first would be the only one to work if "long books" were in any other position.

O3. Possible answers: "I enjoy sleep", topicalized object sentences like "Books I enjoy" and "Sleep I enjoy".



This problem is a follow-up to problem O and has to be solved after that problem. Tok Pisin (also referred to as New Guinea Pidgin or Melanesian Pidgin) is a creole language spoken in the northern mainland of Papua New Guinea and surrounding islands. It is an official language and the mostly widely used language in the country, spoken by over 5 million people.

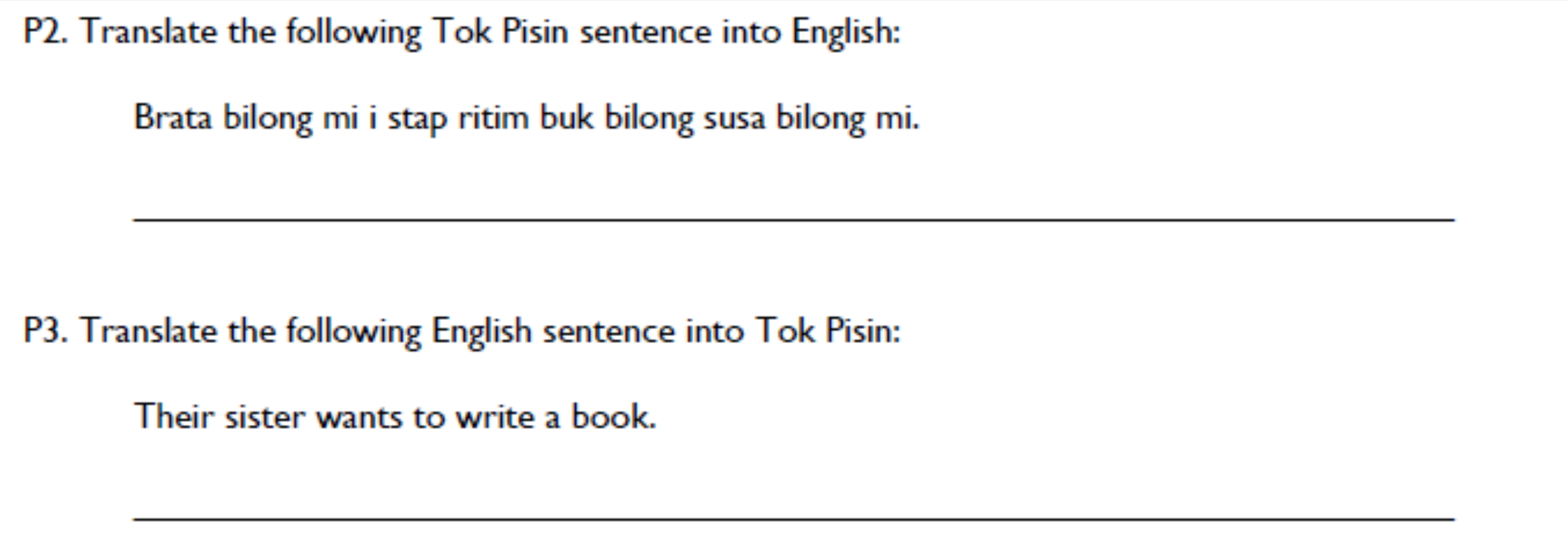
Many Tok Pisin words come originally from English – its name comes from "talk" and "pidgin"[1] -- but Tok Pisin isn't just English. It has a distinct grammar and uses these words in different (but systematic!) ways.

P1. Below are sentences in Tok Pisin with a scrambled list of English translations. Match each sentence to its English equivalent.

1.	Brata bilong em i stap rit.	
2.	Ol i stap dringim wara.	
3.	Ol i ken ritim buk bilong mi.	
4.	Em i ritim buk pinis.	
5.	Em i laik rit.	
6.	Susa bilong em i ken rait.	
7.	Susa bilong mi i boilim wara.	
8.	Wara i boil pinis.	

A.	He has read the book.
B.	My sister boils the water.
C.	They can read my book.
D.	His sister can write.
E.	His brother is reading.
F.	The water has boiled.
G.	He wants to read.
H.	They are drinking water.

CCG



P2. Translate the following Tok Pisin sentence into English:

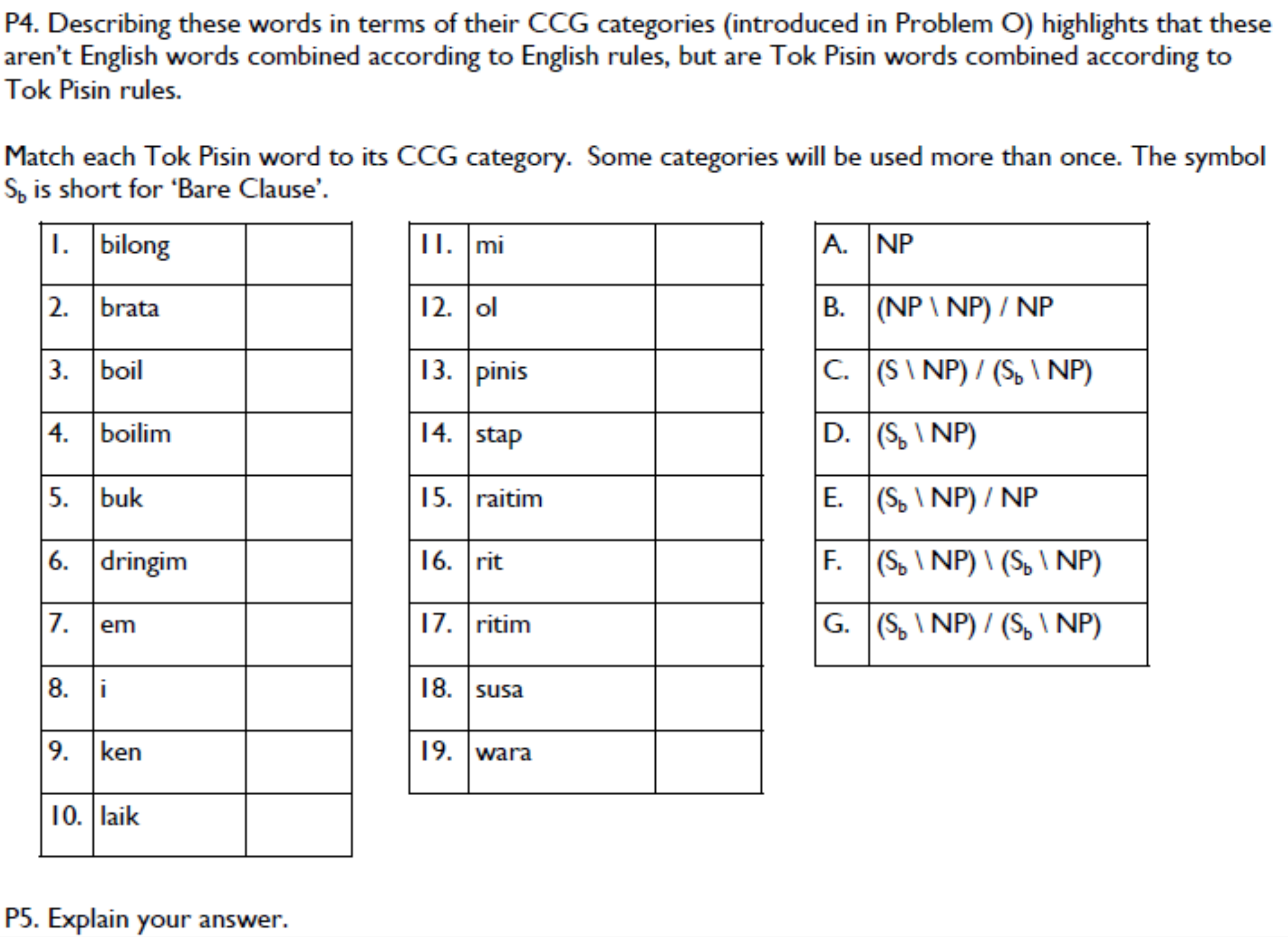
Brata bilong mi i stap ritim buk bilong susa bilong mi.

____

P3. Translate the following English sentence into Tok Pisin:

Their sister wants to write a book.

____



P4. Describing these words in terms of their CCG categories (introduced in Problem O) highlights that these aren't English words combined according to English rules, but are Tok Pisin words combined according to Tok Pisin rules.

Match each Tok Pisin word to its CCG category. Some categories will be used more than once. The symbol S_b is short for 'Bare Clause'.

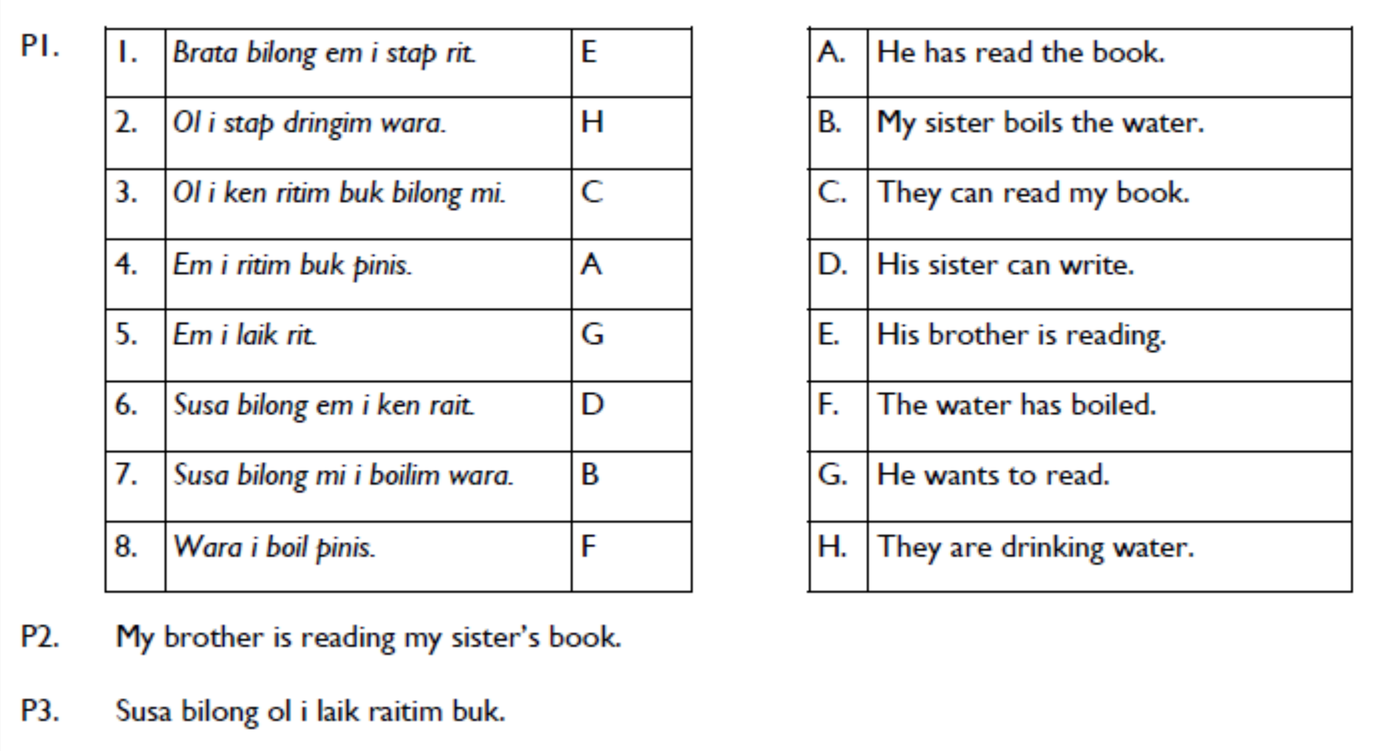
1.	bilong	
2.	brata	
3.	boil	
4.	boilim	
5.	buk	
6.	dringim	
7.	em	
8.	i	
9.	ken	
10.	laik	

11.	mi	
12.	ol	
13.	pinis	
14.	stap	
15.	raitim	
16.	rit	
17.	ritim	
18.	susa	
19.	wara	

A.	NP
B.	(NP \ NP) / NP
C.	(S \ NP) / (S_b \ NP)
D.	(S_b \ NP)
E.	(S_b \ NP) / NP
F.	(S_b \ NP) \ (S_b \ NP)
G.	(S_b \ NP) / (S_b \ NP)

P5. Explain your answer.

CCG



P1.

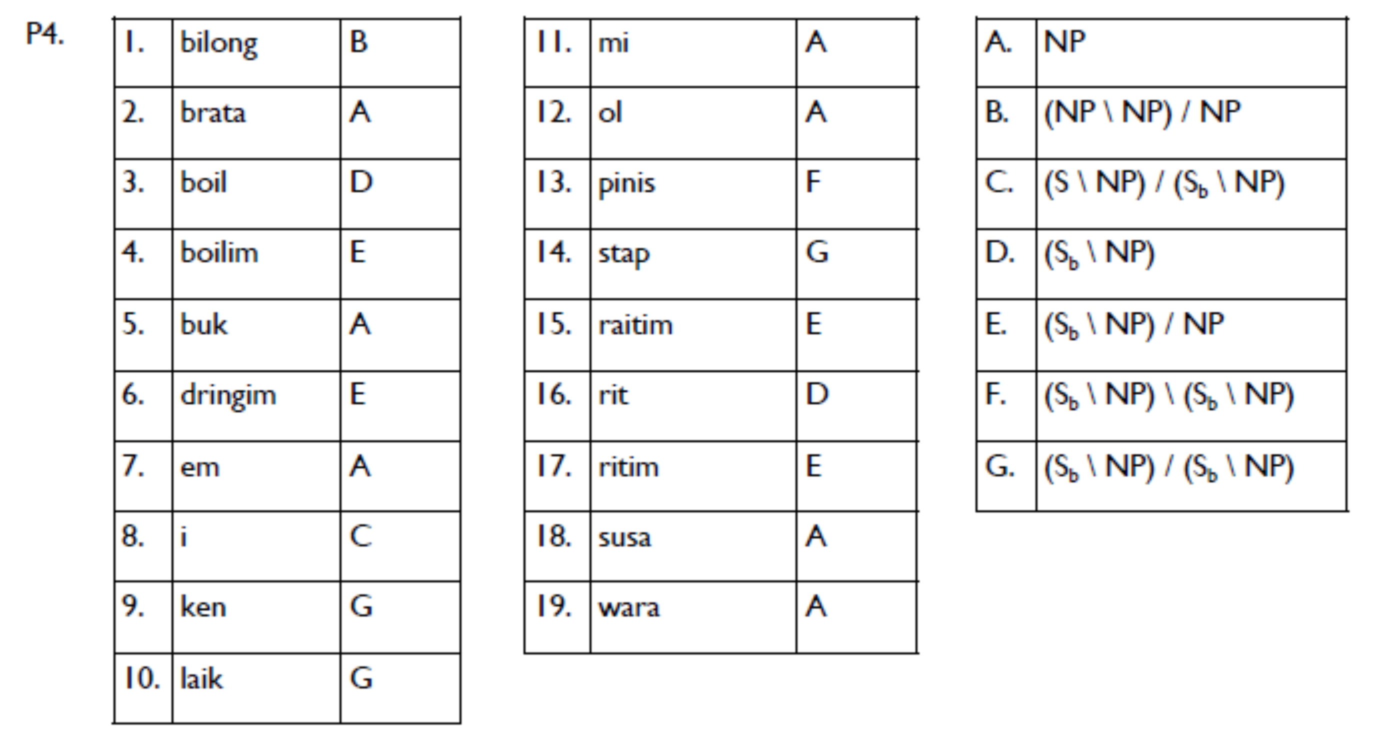
1.	*Brata bilong em i stap rit.*	E
2.	*Ol i stap dringim wara.*	H
3.	*Ol i ken ritim buk bilong mi.*	C
4.	*Em i ritim buk pinis.*	A
5.	*Em i laik rit.*	G
6.	*Susa bilong em i ken rait.*	D
7.	*Susa bilong mi i boilim wara.*	B
8.	*Wara i boil pinis.*	F

A.	He has read the book.
B.	My sister boils the water.
C.	They can read my book.
D.	His sister can write.
E.	His brother is reading.
F.	The water has boiled.
G.	He wants to read.
H.	They are drinking water.

P2. My brother is reading my sister's book.

P3. Susa bilong ol i laik raitim buk.

CCG



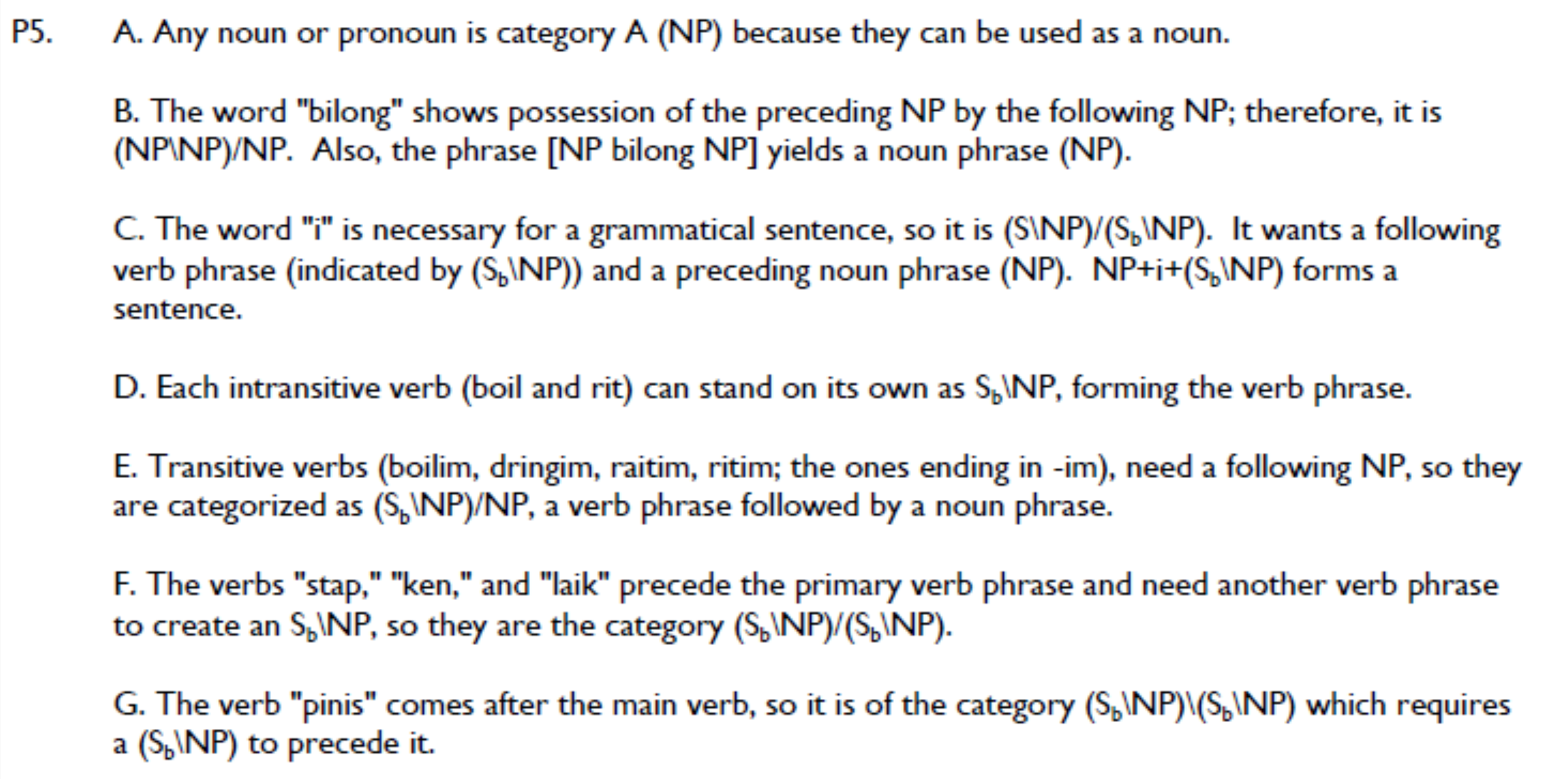
P4.

1.	bilong	B
2.	brata	A
3.	boil	D
4.	boilim	E
5.	buk	A
6.	dringim	E
7.	em	A
8.	i	C
9.	ken	G
10.	laik	G

11.	mi	A
12.	ol	A
13.	pinis	F
14.	stap	G
15.	raitim	E
16.	rit	D
17.	ritim	E
18.	susa	A
19.	wara	A

A.	NP
B.	$(NP \backslash NP) / NP$
C.	$(S \backslash NP) / (S_b \backslash NP)$
D.	$(S_b \backslash NP)$
E.	$(S_b \backslash NP) / NP$
F.	$(S_b \backslash NP) \backslash (S_b \backslash NP)$
G.	$(S_b \backslash NP) / (S_b \backslash NP)$

CCG



P5. A. Any noun or pronoun is category A (NP) because they can be used as a noun.

B. The word "bilong" shows possession of the preceding NP by the following NP; therefore, it is (NP\NP)/NP. Also, the phrase [NP bilong NP] yields a noun phrase (NP).

C. The word "i" is necessary for a grammatical sentence, so it is $(S\backslash NP)/(S_b\backslash NP)$. It wants a following verb phrase (indicated by $(S_b\backslash NP)$) and a preceding noun phrase (NP). NP+i+$(S_b\backslash NP)$ forms a sentence.

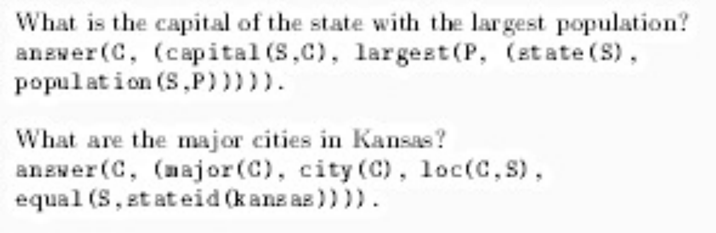
D. Each intransitive verb (boil and rit) can stand on its own as $S_b\backslash NP$, forming the verb phrase.

E. Transitive verbs (boilim, dringim, raitim, ritim; the ones ending in -im), need a following NP, so they are categorized as $(S_b\backslash NP)/NP$, a verb phrase followed by a noun phrase.

F. The verbs "stap," "ken," and "laik" precede the primary verb phrase and need another verb phrase to create an $S_b\backslash NP$, so they are the category $(S_b\backslash NP)/(S_b\backslash NP)$.

G. The verb "pinis" comes after the main verb, so it is of the category $(S_b\backslash NP)\backslash(S_b\backslash NP)$ which requires a $(S_b\backslash NP)$ to precede it.

GeoQuery (Zelle and Mooney 1996)

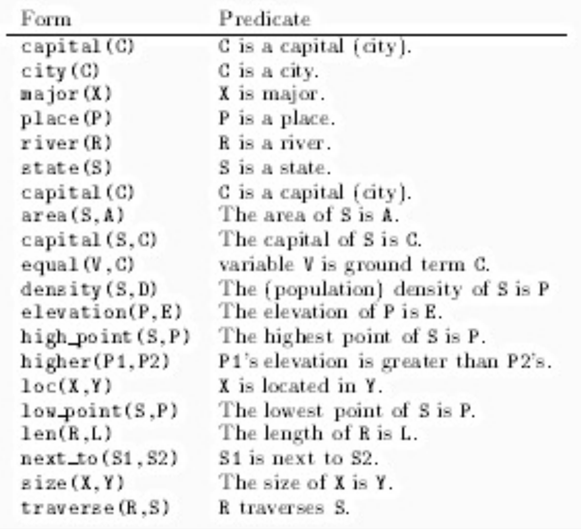


What is the capital of the state with the largest population?

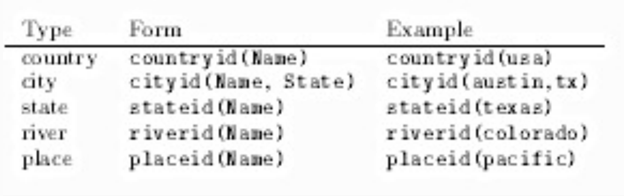
```
answer(C, (capital(S,C), largest(P, (state(S),
population(S,P))))).
```

What are the major cities in Kansas?

```
answer(C, (major(C), city(C), loc(C,S),
equal(S,stateid(kansas)))).
```

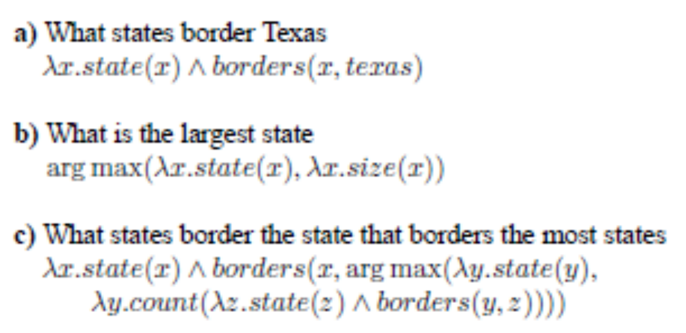


Form	Predicate
capital(C)	C is a capital (city).
city(C)	C is a city.
major(X)	X is major.
place(P)	P is a place.
river(R)	R is a river.
state(S)	S is a state.
capital(C)	C is a capital (city).
area(S,A)	The area of S is A.
capital(S,C)	The capital of S is C.
equal(V,C)	variable V is ground term C.
density(S,D)	The (population) density of S is P
elevation(P,E)	The elevation of P is E.
high_point(S,P)	The highest point of S is P.
higher(P1,P2)	P1's elevation is greater than P2's.
loc(X,Y)	X is located in Y.
low_point(S,P)	The lowest point of S is P.
len(R,L)	The length of R is L.
next_to(S1,S2)	S1 is next to S2.
size(X,Y)	The size of X is Y.
traverse(R,S)	R traverses S.



Type	Form	Example
country	countryid(Name)	countryid(usa)
city	cityid(Name, State)	cityid(austin,tx)
state	stateid(Name)	stateid(texas)
river	riverid(Name)	riverid(colorado)
place	placeid(Name)	placeid(pacific)

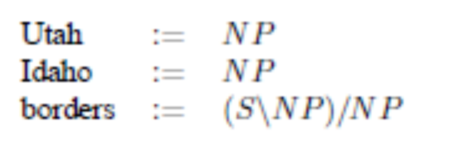
Zettlemoyer and Collins (2005)



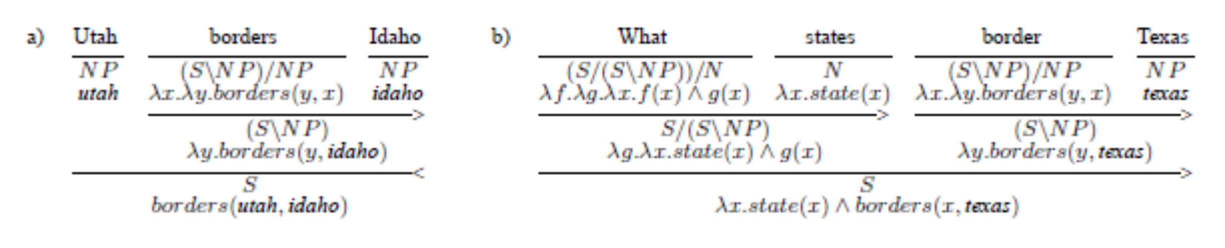
a) What states border Texas
$\lambda x.state(x) \wedge borders(x, texas)$

b) What is the largest state
$\arg\max(\lambda x.state(x), \lambda x.size(x))$

c) What states border the state that borders the most states
$\lambda x.state(x) \wedge borders(x, \arg\max(\lambda y.state(y),$
$\lambda y.count(\lambda z.state(z) \wedge borders(y, z))))$



Utah	:=	NP
Idaho	:=	NP
borders	:=	$(S\backslash NP)/NP$

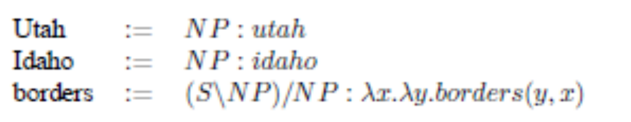


a) Utah borders Idaho

$$\dfrac{\dfrac{\text{Utah}}{\begin{matrix}NP\\ utah\end{matrix}} \quad \dfrac{\dfrac{\text{borders}}{\begin{matrix}(S\backslash NP)/NP\\ \lambda x.\lambda y.borders(y,x)\end{matrix}} \quad \dfrac{\text{Idaho}}{\begin{matrix}NP\\ idaho\end{matrix}}}{\begin{matrix}(S\backslash NP)\\ \lambda y.borders(y, idaho)\end{matrix}}>}{\begin{matrix}S\\ borders(utah, idaho)\end{matrix}}<$$

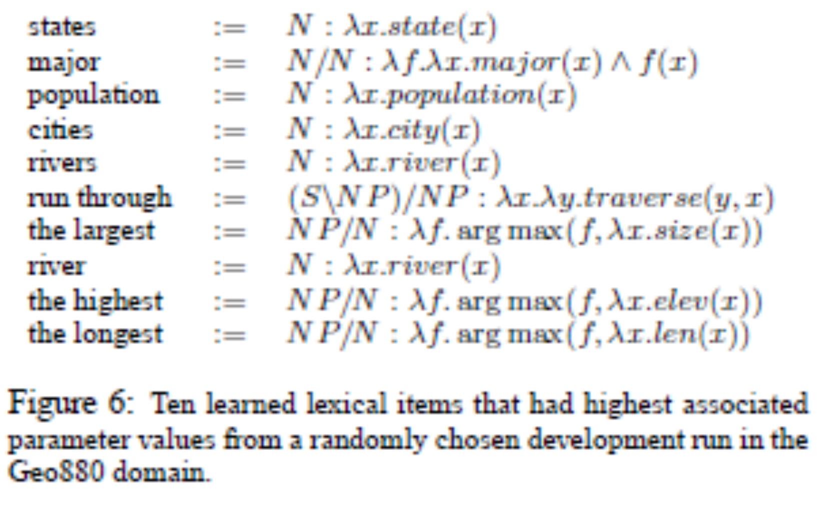
b) What states border Texas

$$\dfrac{\dfrac{\dfrac{\text{What}}{\begin{matrix}(S/(S\backslash NP))/N\\ \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)\end{matrix}} \quad \dfrac{\text{states}}{\begin{matrix}N\\ \lambda x.state(x)\end{matrix}}}{\begin{matrix}S/(S\backslash NP)\\ \lambda g.\lambda x.state(x) \wedge g(x)\end{matrix}}> \quad \dfrac{\dfrac{\text{border}}{\begin{matrix}(S\backslash NP)/NP\\ \lambda x.\lambda y.borders(y,x)\end{matrix}} \quad \dfrac{\text{Texas}}{\begin{matrix}NP\\ texas\end{matrix}}}{\begin{matrix}(S\backslash NP)\\ \lambda y.borders(y, texas)\end{matrix}}>}{\begin{matrix}S\\ \lambda x.state(x) \wedge borders(x, texas)\end{matrix}}>$$



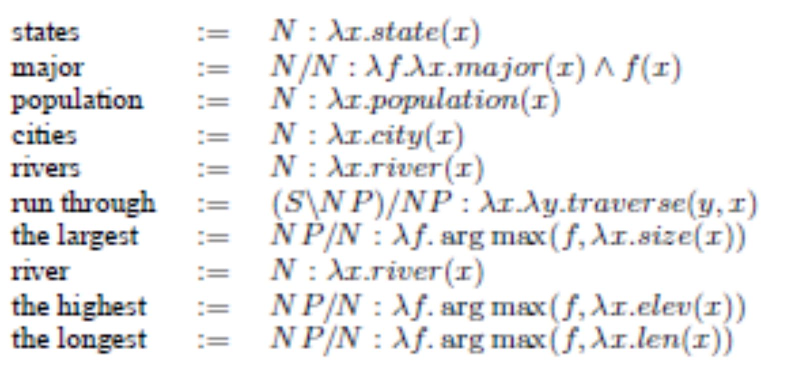
Utah	:=	$NP : utah$
Idaho	:=	$NP : idaho$
borders	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$

Zettlemoyer and Collins (2005)



states	:=	$N : \lambda x.state(x)$
major	:=	$N/N : \lambda f.\lambda x.major(x) \wedge f(x)$
population	:=	$N : \lambda x.population(x)$
cities	:=	$N : \lambda x.city(x)$
rivers	:=	$N : \lambda x.river(x)$
run through	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.traverse(y, x)$
the largest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.size(x))$
river	:=	$N : \lambda x.river(x)$
the highest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.elev(x))$
the longest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.len(x))$

Figure 6: Ten learned lexical items that had highest associated parameter values from a randomly chosen development run in the Geo880 domain.



states	:=	$N : \lambda x.state(x)$
major	:=	$N/N : \lambda f.\lambda x.major(x) \wedge f(x)$
population	:=	$N : \lambda x.population(x)$
cities	:=	$N : \lambda x.city(x)$
rivers	:=	$N : \lambda x.river(x)$
run through	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.traverse(y, x)$
the largest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.size(x))$
river	:=	$N : \lambda x.river(x)$
the highest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.elev(x))$
the longest	:=	$NP/N : \lambda f. \arg\max(f, \lambda x.len(x))$

NLP