University of Toronto, Department of Computer Science
CSC 485H/2501H: Computational linguistics, Fall 2022

Assignment 3

Due date: 23:59 on [illegible], 2022.
Late assignments wil[illegible]out a valid medical certificate or other documentation of an emergency.
For CSC485 students[illegible]orth 33% of your final grade.
For CSC2501 students, this assignment is worth 25% of your final grade.

- **Read the whole assignment carefully.**
- What you turn in must be your own work. You may not work with anyone else on any of the problems in this assignment. If you need assistance, contact the instructor or TA for the assignment.
- Any clarifications to the problems will be posted on the Piazza forum for the class. You will be responsible for taking into account in your solutions any information that is posted there, or discussed in class, so you should check the page regularly between now and the due date.
- The starter code directory for this assignment is accessible on Teaching Labs machines at the path `/u/csc485h/fall/pub/a3/`. In this handout, code files we refer to are located in that directory.
- When implementing code, make sure to **read the docstrings** as some of them provide important instructions, implementation details, or hints.
- Fill in your name, student number, and UTORid on the relevant lines at the top of each file that you submit. (Do not add new lines; just replace the `NAME`, `NUMBER`, and `UTORid` placeholders.)

Overview: Symbolic Machine Translation

In this assignment[illegible] write phrase structure grammars for some different linguistic phenomena in [illegible]es: English and Chinese. You can use the two grammars to create an *interlingu[illegible]n system* by parsing in one, and generating in the other. Don't panic if you do[illegible]d also don't cheer up yet if you can speak the language — it won't give you n[illegible] over other students. A facility with languages in general will help you, as will [illegible]d understand the nuances between the grammars of two different languages. In particular, you will start by working on agreement. Then, you will need to analyse the quantifier scoping difference between the two languages.

TRALE Instructions The TRALE system can be run with:

```
/h/u2/csc485h/fall/pub/trale/trale -fsug
```

(which you are welcome to alias). For this assignment, TRALE needs to start a graphical interface: Gralej. Therefore, if you don't have access to the labs and want to run TRALE remotely, you can either use:

- RDP over SSH (https://www.teach.cs.toronto.edu/using_cdf/rdp.html),
- Remote Access Server NX (https://www.teach.cs.toronto.edu/using_cdf/remote_access_server.html),
- or connect to `teach.cs` using ssh with either the `-X` or `-Y` flag:
  `ssh -X myutorid@teach.cs.toronto.edu`

1. Agreement: Determiners, Numbers and Classifiers [10 marks]

English expresses sub[illegible]n person and number. English has two kinds of number: singular and plural. [illegible]se must agree with its predicate: they should be both singular or both plura[illegible]r of a direct object does not need to agree with anything.

(1) A linguist [illegible]

(2) Two lingui[illegible]

(3) * Two linguists annoys two dolphins.

(4) * A linguist annoy two dolphins.

Chinese, on the other hand, does not exhibit subject-verb agreement. As shown in the examples below, most nouns do not inflect at all for plurality. Chinese does, however, have a classifier (CL) part of speech that English does not. Semantically, classifiers are similar to English collective nouns (a *bottle* of water, a *murder* of crows), but English collective nouns are only used when describing collectives. With very few exceptions, classifiers are mandatory in complex Chinese noun phrases. Different CLs agree with different classes of nouns that are sorted by mostly semantic criteria. For example, 语言学家 (yu yan xue jia)[1] *linguist* is a person and an occupation, so it should be classified by either 个 (ge) or 位 (wei) and cannot be classified by the animal CL 只 (zhi). However, the rules of determining a noun's class constitute a formal system that must be followed irrespective of semantic similarity judgements. For example, while mice and sheep are both animals and can both be classified by the animal CL 只 (zhi), 羊 (yang) *sheep* can take another classifier, 头 (tou), for livestock.

(5) 一 个 语言学家
yi ge yu yan xue jia
one *ge*-CL linguist

(6) 两 个 语言学家
liang ge yu yan xue jia
two *ge*-CL linguist

(7) 三 个 语言学家
san ge yu yan xue jia
three *ge*-CL linguist

(8) * 三 语言学家
san yu yan xue jia
three linguist

(9) * 三 只 语言学家
san zhi yu yan xue jia
three *zhi*-CL linguist

(10) 一 只 老鼠
yi zhi laoshu
one *zhi*-CL mouse

(11) 两 只 老鼠
liang zhi laoshu
two *zhi*-CL mouse

(12) 三 只 老鼠
san zhi laoshu
three *zhi*-CL mouse

(13) * 三 头 老鼠
san *tou*-CL laoshu
three mouse

(14) * 三 位 老鼠
san wei laoshu
three *wei*-CL mouse

[1] Use either Chinese characters or the Romanized form, but with no spaces or hyphens, e.g., yuyanxuejia, for multi-character lexical entries.

You should be familiar by now with the terminology in the English grammar starter code for this question. The Chinese grammar is fairly similar, but there is a new phrasal category called a classifier phrase (CLP), formed by a number and a classifier. The classifier phrase serves the same role as a determiner does in English.

The two grammar riately constrain the NPs generated. You need to design your own rules and fe orce agreement.

English Gr	**inese Grammar**:
Rules:	**les**:
S → NP VP	→ NP VP
VP → V NP	→ V NP
NP → Det N	NP → CLP N
NP → Num N	CLP → Num CL

Lexicon:		**Lexicon**:		
a:	Det	一	yi *one/a*	Num
one:	Num	两	liang *two*	Num
two:	Num	三	san *three*	Num
three:	Num	老鼠	laoshu *mouse*	N
mouse:	N	羊	yang *sheep*	N
mice:	N	语言学家	yu yan xue jia *linguist*	N
sheep:	N	看见	kanjian *see*	V
sheep:	N	追	zhui *chase*	V
linguist:	N	个	ge	CL
linguists:	N	位	wei	CL
see:	V	只	zhi	CL
sees:	V	头	tou	CL
saw:	V			
chase:	V			
chases:	V			
chased:	V			

Here is a list of all of the nouns in this question and their acceptable classifiers:

- 老鼠 laoshu *mouse*: 只 zhi;
- 羊 yang *sheep*: 只 zhi, 头 tou;
- 语言学家 yu yan xue jia *linguist*: 个 ge, 位 wei.

(a) (6 marks) Implement one grammar for each language pursuant to the specifications above. English: q1_en.pl and Chinese: q1_zh.pl.

Neither of your grammars need to handle embedded clauses, e.g., *a linguist saw two mice chase a sheep*. Similarly for Chinese, your grammar doesn't need to parse sentences like example (15):

(15) 一个 i [illegible] 两 只 老鼠 追 一头 羊
yi ge y [illegible] liang zhi laoshu zhui yi tou yang
A lingu [illegible] se a sheep.

For the Chines [illegible] l entries can be coded in either pinyin (the Romanized transcriptions of the Chinese characters) or in simplified Chinese characters.

(b) (4 marks) Use your grammars to parse and translate the following sentences. Save and submit all the translation results in the grale format. The results of sentence (16) should be named q1b_en.grale and the results of sentence (17) should be named q1b_zh.grale.

(16) Two mice chase one sheep.

(17) 一个 语言学家 追 两 头 羊
yi ge yu yan xue jia zhui liang tou yang

Operational Instructions

- If you decide to use simplified Chinese characters, enter them in Unicode and use the -u flag when you run TRALE.
- Independently test your grammars in TRALE first, before trying to translate.
- Use the function translate to generate a semantic representation of your source sentence. If your sentence can be parsed, the function translate should open another gralej interface with all of the translation results.

```
| ?- translate([two,mice,chase,one,sheep]).
```

- To save the translation results, on the top left of the Gralej window (the window with the INITIAL CATEGORY entry and all of the translated sentences listed), click File >> Save all >> TRALE format.
- Don't forget to **close all of the windows** or kill both of the Gralej processes after you finish. Each Gralej process will take up one port in the server, and no one can use the server if we run out of ports.

2. Quantifier Scope [30 marks]

For this assignment, [illegible] quantifiers: the universal quantifier (*every*, 每 mei) and the existential quantif[illegible]ish, both quantifiers behave as singular determiners.

(18)   A professo[illegible]

(19) * A professo[illegible]

(20) * A professo[illegible]

In Chinese, both of these quantifiers behave more like numerical determiners. In addition, when a universal quantifier modifies an NP that occurs before the verb (such as with a universally quantified subject), the preverbal operator 都 (dou) is required. When a universally quantified NP occurs after the verb, the dou-operator must not appear with it.

(21)   Every professor stole a cookie.

(22)   A professor stole every cookie.

(23)   每 个 教授 都 偷了 一 块 饼干
mei ge jiaoshou *dou* tou-le yi kuai binggan
∀ *ge*-CL professor DOU stole ∃ *kuai*-CL cookie

(24) * 每 个 教授 偷了 一 块 饼干
mei ge jiaoshou tou-le yi kuai binggan
∀ *ge*-CL professor stole ∃ *kuai*-CL cookie

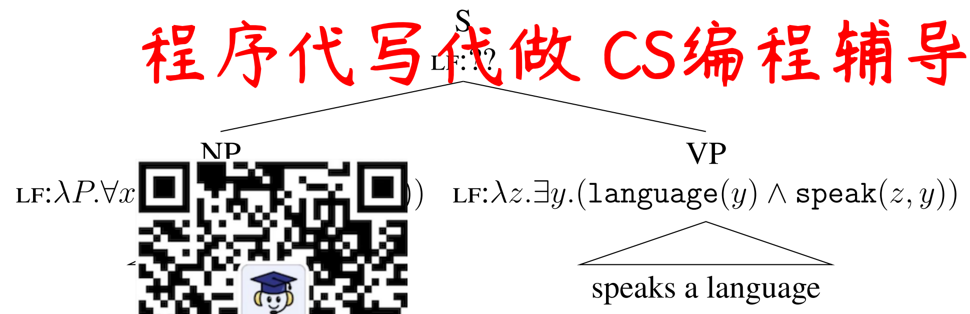
(25)   一 个 教授 偷了 每 块 饼干
yi ge jiaoshou tou-le mei kuai binggan
∃ *ge*-CL professor stole ∀ *kuai*-CL cookie

(26) * 一 个 教授 都 偷了 每 块 饼干
yi ge jiaoshou *dou* tou-le mei kuai binggan
∃ *ge*-CL professor *dou* stole ∀ *kuai*-CL cookie

Quantifier Scope Ambiguity In lecture, we talked about different kinds of ambiguity. In many English sentences, no matter what the order of the quantifiers, there is a quantifier scope ambiguity. For example, the sentence *every hacker speaks a language* has two readings:

- (∃ > ∀) Every hacker speaks a language. [The language is Inuktitut.]
- (∀ > ∃) Every hacker speaks a language. [Some hackers speak Inuktitut and some hackers speak Aymara.]

The symbol (∃ > ∀) means the existential quantifier outscopes the universal quantifier in a logical form representation of the sentence.


S
LF: ??
NP
LF: $\lambda P.\forall x$))
VP
LF: $\lambda z.\exists y.(\texttt{language}(y) \land \texttt{speak}(z, y))$
speaks a language

tion. What should be the LF of S?

We can write the semantics of the two sentences in their logical forms (LF) to distinguish the two readings:

- $\exists y.(\texttt{language}(y) \land \forall x.(\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, y)))$
- $\forall x.(\texttt{hacker}(x) \Rightarrow \exists y.(\texttt{language}(y) \land \texttt{speak}(x, y)))$

English sentences (27, 28) are scopally ambiguous no matter what the linear order of the quantifiers is. But in Chinese, a sentence is scopally ambiguous only when the universally quantified NP precedes the existential NP: (29) is ambiguous, but (30) is unambiguous.[2]

(27) Every hacker speaks a language
Ambiguous: $\exists > \forall$, $\forall > \exists$

(28) A hacker speaks every language
Ambiguous: $\exists > \forall$, $\forall > \exists$

(29)

每	个	黑客	都	会说	一	种	语言
mei	ge	heike	dou	huishuo	yi	zhong	yuyan
∀	*ge*-CL	hacker	DOU	speak	∃	*zhong*-CL	language

Ambiguous: $\exists > \forall$, $\forall > \exists$

(30)

一	个	黑客	会说	每	种	语言
yi	ge	heike	huishuo	mei	zhong	yuyan
∃	*ge*-CL	hacker	speak	∀	*zhong*-CL	language

Unambiguous: $\exists > \forall$

How can we derive the LF of the two readings? We use a process called *beta reduction*. Recall the lambda-calculus notation: $\lambda x.x^2$ denotes a function that takes a variable x, and returns the square of its value (x^2). After substituting the value for the bound variable x, we can reduce the function application in the body of the lambda term to a new expression. For example, applying 2 to $\lambda x.x^2$ will get us:

$$\lambda x.x^2(2) = 2^2$$

[2] The actual principles that determine the scopal readings of a Chinese sentence are still an active area of research, but this is obviously beyond the scope of this assignment. You only need to construct an analysis that explains every example mentioned in this assignment.

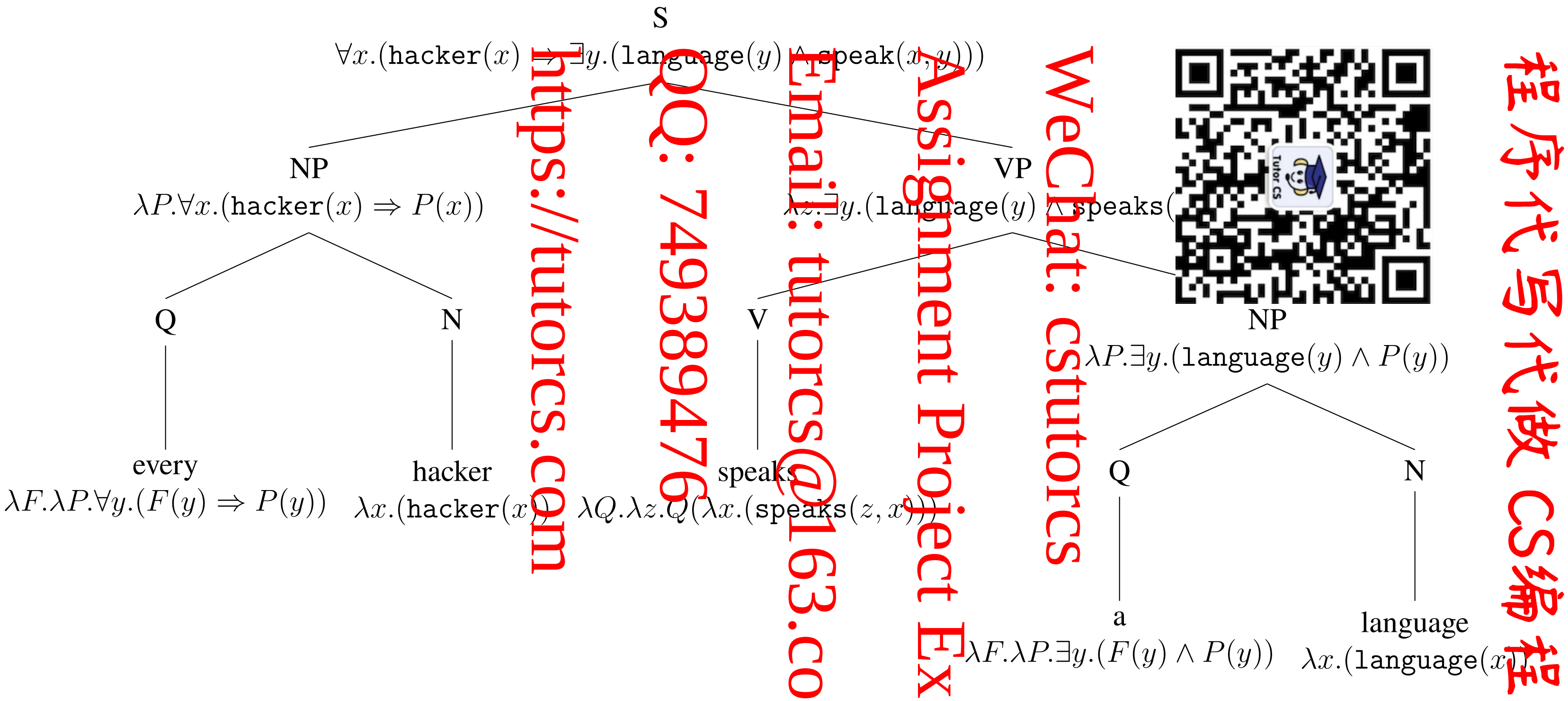

S
$\forall x.(\texttt{hacker}(x) \Rightarrow \exists y.(\texttt{language}(y) \wedge \texttt{speak}(x, y)))$
NP
$\lambda P.\forall x.(\texttt{hacker}(x) \Rightarrow P(x))$
VP
$\lambda x.\exists y.(\texttt{language}(y) \wedge \texttt{speaks}($
Q
N
V
NP
$\lambda P.\exists y.(\texttt{language}(y) \wedge P(y))$
every
$\lambda F.\lambda P.\forall y.(F(y) \Rightarrow P(y))$
hacker
$\lambda x.(\texttt{hacker}(x))$
speaks
$\lambda Q.\lambda z.Q(\lambda x.(\texttt{speaks}(z, x)))$
Q
N
a
$\lambda F.\lambda P.\exists y.(F(y) \wedge P(y))$
language
$\lambda x.(\texttt{language}(x))$

Figure 2: Beta reduction analysis of the sentence *every hacker speaks a language.*

This process is also known as beta reduction (denoted as $\Leftrightarrow_\beta$). Note that beta reduction itself does not tell us that this equals 4. That is obtained by a subsequent process of arithmetic evaluation. But we can use beta reduction even if we don't evaluate.

We can also perform beta reduction on variables for functions. For example, applying in $\lambda F.F(2)$ to $\lambda x.x^2$ will yield:

$$x.x^2) = (\lambda x.x^2)(2) = 2^2$$

Now, let's look a es beta reduction to compute the LF of a sentence. For example, as shown in t the LF of the NP *every hacker* is $\lambda P.\forall x.(\texttt{hacker}(x) \Rightarrow P(x))$ and the LF of t *uage* is $\lambda z.\exists y.(\texttt{language}(y) \land \texttt{speaks}(z, y))$. What is the LF of *every hacke*

$$\begin{aligned} &\lambda P.\forall x.(\texttt{hacker}(x) \Rightarrow P(x))(\lambda z.\exists y.(\texttt{language}(y) \land \texttt{speak}(z, y))) \\ \Leftrightarrow_\beta\ &\forall x.(\texttt{hacker}(x) \Rightarrow \lambda z.\exists y.(\texttt{language}(y) \land \texttt{speak}(z, y))(x)) \\ \Leftrightarrow_\beta\ &\forall x.(\texttt{hacker}(x) \Rightarrow \exists y.(\texttt{language}(y) \land \texttt{speak}(x, y))) \end{aligned}$$

Each step of repeatedly applying beta reduction to every subterm until we reach an irreducible statement is called *beta normalisation*.

Figure 2 shows the complete analysis of the sentence *every hacker speaks a language*. Familiarize yourself with every part of the analysis. But this only generates one of the two readings – the surface reading ($\forall > \exists$). We will use a technique called *quantifier storage* to capture the scopal ambiguity and make both readings available.

Quantifier Storage If quantifier scoping is a semantic effect, how do we represent it in syntax? When there is no ambiguity, keeping track of the quantifier scope is pretty straightforward. To keep track of and resolve scope ambiguities, we will use a list called a *quantifier store*. The idea behind QSTORE is that, instead of consuming all of the LF components right away, we can choose to keep them in QSTORE and apply them later.

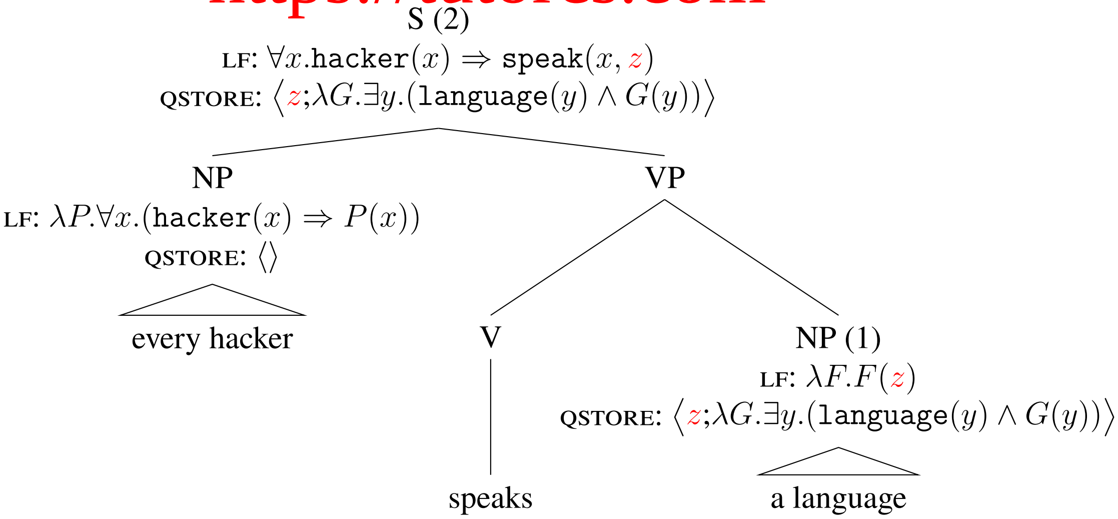

S (2)
LF: $\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z)$
QSTORE: $\langle z;\lambda G.\exists y.(\texttt{language}(y) \land G(y))\rangle$
NP
LF: $\lambda P.\forall x.(\texttt{hacker}(x) \Rightarrow P(x))$
QSTORE: $\langle\rangle$
every hacker
VP
V
speaks
NP (1)
LF: $\lambda F.F(z)$
QSTORE: $\langle z;\lambda G.\exists y.(\texttt{language}(y) \land G(y))\rangle$
a language

Figure 3: Quantifier Storage. Storing the quantifier at (1), and retrieve it later at (2).

Let's go back to the example, *every hacker speaks a language* (figure 3). We first store the LF of the NP *a language* at (1) and replace the LF of the NP with a placeholder $\lambda F.F(z)$. The variable z in this expression is a free occurrence, and it is the same variable as the z in the store and in the LF of the sentence (the free occurrences of z are highlighted in red). We retrieve the logical form from the store at (2). The r[illegible]sts of three steps:

1. First, we constr[illegible] where L_S is the current LF, and z is the variable paired in the QSTORE ent[illegible]ase, this will yield $\lambda z.(\forall x, \texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z))$.

2. Then, we apply[illegible]F from the QSTORE entry.

3. Finally, we bet[illegible]eta normalisation, we obtain the second reading of the sentence.

$$\begin{aligned}
&\lambda G.\exists y.(\texttt{language}(y) \wedge G(y))(\lambda z.(\forall x, \texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z)))\\
\Leftrightarrow_\beta\ &\exists y.(\texttt{language}(y) \wedge (\lambda z.(\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z))(y))\\
\Leftrightarrow_\beta\ &\exists y.(\texttt{language}(y) \wedge (\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, y))
\end{aligned}$$

Topicalization and Movement Topicalization is a linguistic phenomenon in which an NP appears at the beginning of a sentence in order to establish it as the topic of discussion in a sentence or to emphasize it in some other way. It plays an important role in the syntax of fixed-word-order languages because grammatical function is mainly determined by word order. Both Chinese and English exhibit topicalization. The entire object NP, for example, can be moved to the beginning of the sentence in either language. But in Chinese, object topicalization is more restricted when the subject is quantified: it can happen when the subject is universally quantified, but not when it is existentially quantified (31-36).

(31) A language, every hacker speaks.
∃ language ∀ hacker speaks
Ambiguous: ∀ > ∃ and ∃ > ∀

(32) Every language, a hacker speaks.
∀ language ∃ hacker speaks
Ambiguous: ∀ > ∃ and ∃ > ∀

(33) 一 种 语言 每 个 黑客 都 会说
yi zhong yuyan mei ge heike dou huishuo
∃ *zhong*-CL language ∀ *ge*-CL hacker *dou* speak
Unambiguous: ∃ > ∀

(34) 每 种 语言 每 个 黑客 都 会说
mei zhong yuyan mei ge heike dou huishuo
∀ *zhong*-CL language ∀ *ge*-CL hacker *dou* speak

(35) * 一 种 语言 一 个 黑客 会说
yi zhong yuyan yi ge heike huishuo
∃ *zhong*-CL language ∃ *ge*-CL hacker speak

(36) *每 种 语言 一 个 黑客 (都) 会说
mei ben shu yi ge heike (dou) duguo
∀ *ben*-CL language ∃ *ge*-CL hacker *dou* speak

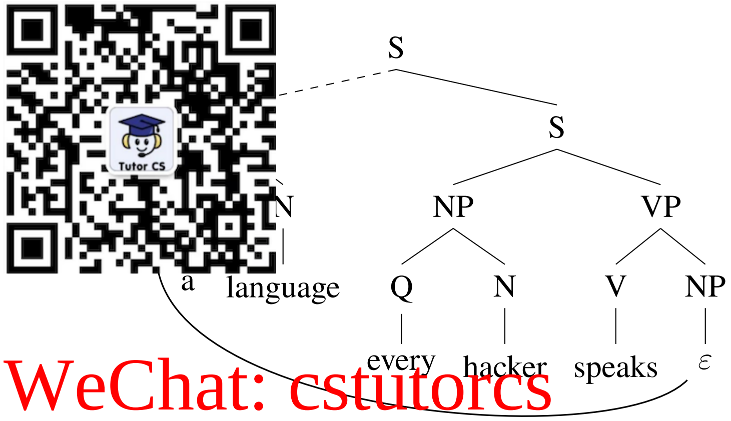


Figure 4: English topicalization parse tree: example (31).

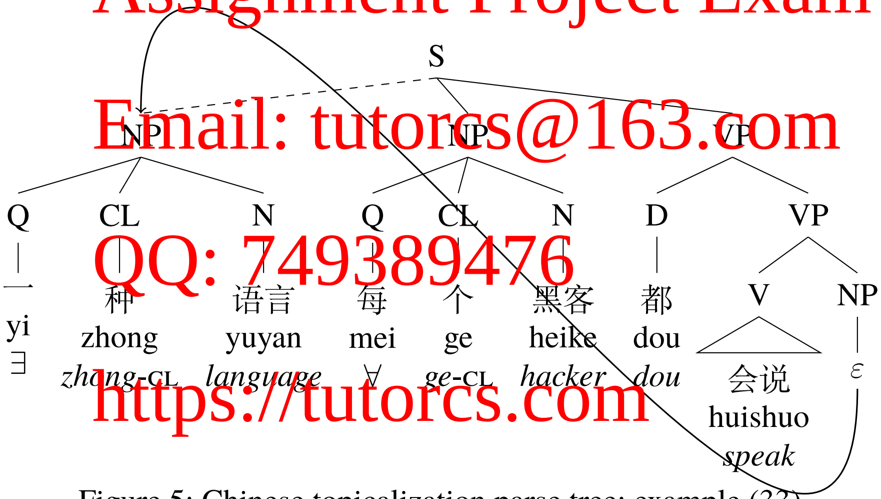


Figure 5: Chinese topicalization parse tree: example (33).

In English, neither subject–verb agreement nor quantifier scope ambiguity is generally affected by movement. In particular, the number and person of the subject should always agree with the predicate no matter where it occurs. Here, you can assume that Chinese also follows subject–verb agreement (regarding the requirement of *dou*) in the same way that English does. But whereas in English, both readings are still available after the sentences are topicalised (31, 32), this is not the case in Chinese. Compared to its untopicalised counterpart (29), the topicalised sentence (33) is no longer ambiguous.

Figures 4 and 5 show the parse trees of sentences (31) and (33). Topicalization is generally analysed with gaps. An empty trace is left in the untopicalized position of the object NP, where the gap is introduced. The gapped NP then percolates up the tree, and is finally unified with the topicalized NP at the left periphery of the sentence.[3]

[3] Although Chinese is an SVO (Subject-Verb-Object) language, there is a means of performing “double movement.”

(a) (2 marks) Manually convert all readings of the sentences (28) and (30) to logical expressions. Put your logical forms in section 2(a) of analysis.txt. Use exists and forall for the quantifiers, and use => and the caret symbol ^ for implication and conjunction.

(b) (10 marks) Imp[illegible] the syntax of quantifier scope ambiguity. You don't need to account for r[illegible]guity in meanings (there should be no syntactic ambiguities). At this p[illegible]ar will produce exactly **one** parse for every grammatical sentence. Test y[illegible] before you move on to the next step.

(c) (10 marks) Au[illegible]s to represent meaning and quantifier scope ambiguity. Marks for ques[illegible]ucted if your work on this part causes errors in the syntactic predictio[illegible]ould generate more than one parse for each ambiguous sentence.

(d) (4 marks) Translate sentences (28) and (30), as you did in the first question.

Operational Instructions

- Use the function translate to generate semantic representations of your source sentences. If your sentences can be parsed, translate should open another gralej window and with all of the translation results.

```
| ?- translate([a,hacker,speaks,every,language]).
```

- You will be prompted as follows to see the next parse.

```
ANOTHER? y
...
ANOTHER? y
no
```

Answer y to see the next parse until you reach the end. Each time TRALE will open a new Gralej window. You need to store all of your translation results by repeating the previous step. A no will be returned when you reach the end of your parses.

- Save your translations of sentence (28) as q2d_28_1.grale, q2d_28_2.grale …and your translations of sentence (30) as q2d_30_1.grale, q2d_30_2.grale …
- Submit a zip file q2d.zip containing all the translation results. You can use this command: zip -r q2d.zip q2d_*.grale to create the zip file.
- Again, don't forget to **close all the windows** and kill your Gralej processes after you finish.

(1)	一 个	黑客	每	种	语言	都	会说
	yi ge	heike	mei	zhong	yuyan	dou	huishuo
	∃ *ge*-CL	hacker	∀	*zhong*-CL	language	*dou*	speak

A hacker every language speak.

We will ignore these.

(e) (4 marks) Compare your translator with Google Translate[4]. At its core, Google Translate is a neural machine translation (NMT) system. In a few sentences, describe the similarities and differences between Google Translate's performance and your system's. Report **at least one** instance of a difference between the translation given by your translator and by Google Translate. Your [illegible]ubmitted as section 2(e) in `analysis.txt`.

[4] https://translate.google.ca/

CSC 485H/2501H, Fall 2022: Assignment 3

Family name: ______________________________

Given name: ______________________________

Student #: ______________________________

Date: ______________________________

I declare that this assignment, both my paper and electronic submissions, is my own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters and the Code of Student Conduct.

Signature: ______________________________