For Question A, I basically follow the grammar given in the handout. For the features in oneb.pl, I developed 3 features for my noun. Is the noun a proper noun? Is the noun a singular or plural? Is it a direct or indirect in the sentence? I considered this three question and added them into my features.

For Question 2, first of all, I started to develop my basic grammar from the sentences in the handout.

The rules I built were:

S -> NP VP

NP -> DET N.

And VP rules with the given verbs.

Writing the lexical entries is an important step for the grammar design as well. We can think about what thematic roles those verbs have. These thematic roles are the important features for the verbs and helped me to fill out the blank in v\_sem.

I customized the features with different verbs. For example, “try” and “expect” have two thematic roles assignable: Agent and Theme; “promise” has three: Agent, Theme and Beneficiary; etc. …

Then, I thought about some other cases which are not covered in the handout.

For example, the sentence “the student expected the teacher to try to sleep” was developed base on the sentence “the student expected the teacher to sleep”. “to try to sleep” is a nested toinf\_clause. I added other cases A1 and A2 rules in grammar to cover this.

However, there are still some limitations in my grammar.

For example:

“The teacher expected {the student tried to sleep} to sleep.”

This is not covered in my grammar for now. I believe there are still many nested grammatical sentences is not cover in my grammar.

**Question 1 A. onea.pl**

% Quan Zhou, zhouqua7, 1002162492

% Type Declaration

bot sub [s, np, npsg, vpsg, nppl, vppl, vsg, vpl, pp, p, det, nprp, nsg, npl].

s sub [].

np sub [].

npsg sub [].

vpsg sub [].

nppl sub [].

vppl sub [].

vsg sub [].

vpl sub [].

pp sub [].

p sub [].

det sub [].

nprp sub [].

nsg sub [].

npl sub [].

% Rule Declaration

s\_rule rule

s ===>

cat> npsg,

cat> vpsg.

s\_rule rule

s ===>

cat> nppl,

cat> vppl.

vpsg\_rule rule

vpsg ===>

cat> vsg,

cat> np.

vppl\_rule rule

vppl ===>

cat> vpl,

cat> np.

pp\_rule rule

pp ===>

cat> p,

cat> np.

npsg\_rule rule

npsg ===>

cat> nprp.

npsg\_rule rule

npsg ===>

cat> det,

cat> nsg.

npsg\_rule rule

npsg ===>

cat> det,

cat> nsg,

cat> pp.

nppl\_rule rule

nppl ===>

cat> det,

cat> npl.

nppl\_rule rule

nppl ===>

cat> det,

cat> npl,

cat> pp.

nppl\_rule rule

nppl ===>

cat> npl.

nppl\_rule rule

nppl ===>

cat> npl,

cat> pp.

np\_rule rule

np ===>

cat> npsg.

np\_rule rule

np ===>

cat> nppl.

% Lexicon Declaration (in alphabetical order)

biscuits ---> npl.

dog ---> nsg.

feed ---> vpl.

feeds ---> vsg.

fido ---> nprp.

puppies ---> npl.

the ---> det.

with ---> p.

**Question 1 A. onea.gralej**

% Quan Zhou, zhouqua7, 1002162492

<'fido feeds the dog with biscuits'

{ :'s\_rule//0:fido feeds the dog with biscuits'

s

{ :'npsg\_rule//0:fido'

npsg

{ :'lexicon:fido'

nprp

}

}

{ :'vpsg\_rule:feeds the dog with biscuits'

vpsg

{ :'lexicon:feeds'

vsg

}

{ :'np\_rule//0:the dog with biscuits'

np

{ :'npsg\_rule//2:the dog with biscuits'

npsg

{ :'lexicon:the'

det

}

{ :'lexicon:dog'

nsg

}

{ :'pp\_rule:with biscuits'

pp

{ :'lexicon:with'

p

}

{ :'np\_rule//1:biscuits'

np

{ :'nppl\_rule//2:biscuits'

nppl

{ :'lexicon:biscuits'

npl

}

}

}

}

}

}

}

}

>

**Question 1 B. oneb.pl**

% Quan Zhou, zhouqua7, 1002162492

bot sub[cat].

cat sub [s,np,vp,p,pp,det].

s sub [].

np sub [] intro [noun:n].

n sub [] intro [n\_prp:n\_prp, sing\_pl:sing\_pl, dir\_indir:dir\_indir].

n\_prp sub [prp, nprp].

prp sub [].

nprp sub [].

sing\_pl sub [sing, plural].

sing sub [].

plural sub [].

dir\_indir sub [direct, indirect].

direct sub [].

indirect sub [].

vp sub [] intro [verb:v].

v sub [] intro [subject:n].

p sub [].

pp sub [].

det sub [].

% Rules Declaration

% S -> NP VP

s\_rule rule

s ===>

cat> (np,noun:sing\_pl:sing\_pl),

cat> (vp,verb:subject:sing\_pl:sing\_pl).

% VP -> V VP

vp\_rule rule

(vp, verb:subject:sing\_pl:sing\_pl) ===>

cat> (v, subject:sing\_pl:sing\_pl),

cat> np.

% PP -> P NP

pp\_rule rule

pp ===>

cat> p,

cat> np.

% NP -> N

np\_rule rule

(np, noun:sing\_pl:sing\_pl) ===>

cat> (n, sing\_pl:sing\_pl, dir\_indir:direct).

% NP -> Det N

np\_det\_rule rule

(np, noun:sing\_pl:sing\_pl) ===>

cat> det,

cat> (n, n\_prp:nprp, sing\_pl:sing\_pl).

% NP -> Det N PP

np\_det\_pp\_rule rule

(np, noun:sing\_pl:sing\_pl) ===>

cat> det,

cat> (n, n\_prp:nprp, sing\_pl:sing\_pl),

cat> pp.

% NP -> N PP

np\_pp\_rule rule

(np, noun:sing\_pl:sing\_pl) ===>

cat> (n, n\_prp:nprp, sing\_pl:sing\_pl),

cat> pp.

% Lexicon Declaration (in alphabetical order)

biscuits ---> (n,n\_prp:nprp,sing\_pl:plural,dir\_indir:direct).

dog ---> (n,n\_prp:nprp,sing\_pl:sing,dir\_indir:indirect).

feed ---> (v, subject:sing\_pl:plural).

feeds ---> (v, subject:sing\_pl:sing).

fido ---> (n,n\_prp:prp,sing\_pl:sing,dir\_indir:direct).

puppies ---> (n, n\_prp:nprp,sing\_pl:plural,dir\_indir:direct).

the ---> det.

with ---> p.

**Question 1 C. onec.gralej**

% Quan Zhou, zhouqua7, 1002162492

<'fido feeds the dog with biscuits'

{ :'s\_rule:fido feeds the dog with biscuits'

s

{ :'np\_rule:fido'

np(

noun: 'mgsat(n)')

{ :'lexicon:fido'

n(

dir\_indir: direct,

n\_prp: prp,

sing\_pl: sing)

}

}

{ :'vp\_rule:feeds the dog with biscuits'

vp(

verb: 'mgsat(v)')

{ :'lexicon:feeds'

v(

subject: n(

dir\_indir: dir\_indir,

n\_prp: n\_prp,

sing\_pl: sing))

}

{ :'np\_det\_pp\_rule:the dog with biscuits'

np(

noun: 'mgsat(n)')

{ :'lexicon:the'

det

}

{ :'lexicon:dog'

n(

dir\_indir: indirect,

n\_prp: nprp,

sing\_pl: sing)

}

{ :'pp\_rule:with biscuits'

pp

{ :'lexicon:with'

p

}

{ :'np\_rule:biscuits'

np(

noun: 'mgsat(n)')

{ :'lexicon:biscuits'

n(

dir\_indir: direct,

n\_prp: nprp,

sing\_pl: plural)

}

}

}

}

}

}

>

**Question 2 B. twob.pl**

% Quan Zhou, zhouqua7, 1002162492

:- ale\_flag(subtypecover,\_,off).

:- discontiguous sub/2,intro/2.

bot sub [mood, tense, sem, cat, pos, verbal, nominal].

% parts of speech

pos sub [n,p,v,det,toinf].

n sub [].

v sub [].

p sub [].

det sub [].

toinf sub []. % infinitival to

% phrasal categories

cat sub [vproj,np].

vproj sub [inf\_clause,s,vp] intro [mood:mood].

inf\_clause intro [mood:infinitive].

s intro [mood:indicative].

vp intro [mood:indicative].

np sub [].

verbal sub [v,vproj] intro [vsem:v\_sem].

nominal sub [n,np] intro [nsem:n\_sem].

% mood and tense for verbs

tense sub [past, present].

past sub [].

present sub [].

mood sub [indicative,infinitive].

indicative intro [tense:tense].

infinitive sub [].

% semantics for verbs and nouns

sem sub [v\_sem, n\_sem].

% semantics for nouns

n\_sem sub [student, teacher].

student sub [].

teacher sub [].

% semantics for verbs

v\_sem sub [try, appear, promise, expect, sleep]

intro [vform:tense, agent:nsem\_none, theme:nsem\_none, beneficiary:nsem\_none, experiencer:nsem\_none, passtype:type].

nsem\_none sub [n\_sem, none].

none sub [].

type sub [object, subject, none].

object sub [].

subject sub [].

% This should not be empty! Fill in features for this and

% the following subtypes:

% try: Agent, Theme %

try sub [] intro [vform:tense, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none].

% appear: Theme %

appear sub [] intro [vform:tense, agent:none, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none].

% promised: Agent, Theme, Beneficiary %

promise sub [] intro [vform:tense, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject].

% expected: Agent, Theme %

expect sub [] intro [vform:tense, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object].

% sleep: Experiencer %

sleep sub [] intro [vform:tense, agent:none, theme:none, beneficiary:none, experiencer:n\_sem, passtype:none].

% ========================= Rules Declaration. =========================%

% the student slept (good) | the student sleep (bad)

s\_rule rule

s

===>

cat> (np, nsem:n\_sem),

cat> (vp, vsem:(vform:past)).

% the student/teacher slept(good) | student slept (bad)

np\_rule rule

np

===>

cat> det,

cat> n.

% --------------------------------------------------------------------------------------------------

% try: Agent, Theme // the student tried {to sleep}.

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> (inf\_clause, mood:infinitive).

% other cases A1: the student tried [to PROMISE the teacher to sleep].

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> toinf,

cat> (v, vsem:(vform:present, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject)),

cat> (np, nsem:n\_sem),

cat> (inf\_clause, mood:infinitive).

% other cases A2: the student tried [to EXPECT the teacher to sleep].

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> toinf,

cat> (v, vsem:(vform:present, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object)),

cat> (np, nsem:n\_sem),

cat> (inf\_clause, mood:infinitive).

% --------------------------------------------------------------------------------------------------

% appear: Theme // the student appeared {to sleep}.

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:none, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> (inf\_clause, mood:infinitive).

% --------------------------------------------------------------------------------------------------

% promise: Agent, Theme, Beneficiary // the student promised the teacher {to sleep}.

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject)),

cat> (np, nsem:n\_sem),

cat> (inf\_clause, mood:infinitive).

% other cases B: the student PROMISED the teacher [to try to sleep]. % =========================>>>

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject)),

cat> (np, nsem:n\_sem),

cat> toinf,

cat> (v, vsem:(vform:present, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> (inf\_clause, mood:infinitive).

% --------------------------------------------------------------------------------------------------

% expected: Agent, Theme // the student expected the teacher {to sleep}.

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object)),

cat> (np, nsem:n\_sem),

cat> (inf\_clause, mood:infinitive).

% other cases B: the student EXPECTED the teacher [to try to sleep]. % =========================>>>

vp\_rule rule

(vp, vsem:(vform:past))

===>

cat> (v, vsem:(vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object)),

cat> (np, nsem:n\_sem),

cat> toinf,

cat> (v, vsem:(vform:present, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none)),

cat> (inf\_clause, mood:infinitive).

% --------------------------------------------------------------------------------------------------

% sleep: Experiencer // the student slept.

vp\_rule rule

(vp, vsem:(vform:past, experiencer:n\_sem))

===>

cat> (v, vsem:(vform:past, theme:none, experiencer:n\_sem)).

% --------------------------------------------------------------------------------------------------

% to sleep

toinf\_rule rule

(inf\_clause, mood:infinitive)

===>

cat> toinf,

cat> (v, vsem:(vform:present, experiencer:n\_sem)).

% --------------------------------------------------------------------------------------------------

% ========================= Lexicon Declaration. (in alphabetical order) =========================%

% appear: Theme

appear ---> (v, vsem:(appear, vform:present, agent:none, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none)).

appeared ---> (v, vsem:(appear, vform:past, agent:none, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none)).

% expected: Agent, Theme

expect ---> (v, vsem:(expect, vform:present, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object)).

expected ---> (v, vsem:(expect, vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:object)).

% promised: Agent, Theme, Beneficiary

promise ---> (v, vsem:(promise, vform:present, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject)).

promised ---> (v, vsem:(promise, vform:past, agent:n\_sem, theme:n\_sem, beneficiary:n\_sem, experiencer:none, passtype:subject)).

% sleep: Experiencer

sleep ---> (v, vsem:(sleep, vform:present, agent:none, theme:none, beneficiary:none, experiencer:n\_sem, passtype:none)).

slept ---> (v, vsem:(sleep, vform:past, agent:none, theme:none, beneficiary:none, experiencer:n\_sem, passtype:none)).

% the others

student ---> (n, nsem:student).

teacher ---> (n, nsem:teacher).

the ---> det.

% toinf

to ---> toinf.

% try: Agent, Theme

try ---> (v, vsem:(try, vform:present, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none)).

tried ---> (v, vsem:(try, vform:past, agent:n\_sem, theme:n\_sem, beneficiary:none, experiencer:none, passtype:none)).

**Question 2 C. twoc.gralej**

% Quan Zhou, zhouqua7, 1002162492

<'the student appeared to sleep'

{ :'s\_rule:the student appeared to sleep'

s(

mood: indicative(

tense: tense),

vsem: 'mgsat(v\_sem)')

{ :'np\_rule:the student'

np(

nsem: n\_sem)

{ :'lexicon:the'

det

}

{ :'lexicon:student'

n(

nsem: student)

}

}

{ :'vp\_rule//3:appeared to sleep'

vp(

mood: indicative(

tense: tense),

vsem: v\_sem(

agent: nsem\_none,

beneficiary: nsem\_none,

experiencer: nsem\_none,

passtype: type,

theme: nsem\_none,

vform: past))

{ :'lexicon:appeared'

v(

vsem: appear(

agent: none,

beneficiary: none,

experiencer: none,

passtype: none,

theme: n\_sem,

vform: past))

}

{ :'toinf\_rule:to sleep'

inf\_clause(

mood: infinitive,

vsem: 'mgsat(v\_sem)')

{ :'lexicon:to'

toinf

}

{ :'lexicon:sleep'

v(

vsem: sleep(

agent: none,

beneficiary: none,

experiencer: n\_sem,

passtype: none,

theme: none,

vform: present))

}

}

}

}

>