**CSC485 Assignment2 Report**

# **Part 1**

**1. Design**

For grammar1, since it's very simple, I just implemented the entire programming based on the given grammar rules and vocabulary. By carefully comparing grammar1 and grammar2, we can see that grammar2 is a simplified version of grammar1 in terms of grammar rules, which means we need to find the difference between them to make each vocabulary bring its own features. The main difference lies in the singular and plural forms of nouns and the singular and plural forms of verbs as well as proper nouns. I carefully designed the features and finally realized the function.

**2. Limitation**

Since the grammar rules of grammar1 and grammar2 are very simple, there are bound to be many restrictions, for example it can't parse different tense (like "Fido fed the dog with biscuits"). Also, it has some overgenerations. Because my grammar is context-free, it may parse some meaningless sentences correctly. (like “The dog feeds the dog with the dog”).

**3. Test Strategy**

I test these sentences to make the grammar work normally.

1). “fido feeds the dog with biscuits”. Of course, this is the given sentence and it must be parse correctly.

2). “fido feeds the dog”. This sentence is to test NPs without PP and it successes.

3). “fido feeds puppies with the dog with biscuits”. Testing this sentence is to make sure PP 🡪 P NP and NPpl 🡪 Npl PP work well.

4). “biscuits feed the dog with fido”. Though this sentence makes no sense, I just to confirm subject-verb agreement (S 🡪 NPsg VPsg and S 🡪 NPpl VPpl).

# **Part 2**

**1. Design**

According to the requirements, I design six features for each verb, including vform, agent, theme, ben:, exp, gap\_type. Vform refers to the tense of the verb, ben refers to Beneficiary, exp refers to Experiencer, and gap\_type refers to whether sleep belongs to subject or object. At the same time, I also design two feature (gap and rec) to save and transmit subject and object. I design one vp rule for every sentence without adding extra nonterminals.

**2. Limitation**

First of all, because I didn't do too much processing on nouns, my grammar can’t parse the noun phase like “students” or “teachers”. Also, it can’t parse the present tense or other tense of verb like “the student try to sleep”. As the same, due to the context-free feature, my grammar may parse some meaningless sentences (like “The teacher promised the teacher to sleep.”).

**3. Test Strategy**

I firstly test the sentence (“the teacher slept”, success) to confirm my grammar and features work well. Then I test all the sentences given in the assignment to make sure my grammar could meet the basic requirement and all the roles of verbs are assigned correctly. Also , I test whether the transitive verb and the intransitive verb can be distinguished well (“the student promised to sleep” and “the student tried the teacher to sleep”, all fail). At the same time, I test the noun phase (“student appeared the teacher to sleep”, fail). Of course, I don’t pay more attention to the noun phase so it just meet the rule (NP 🡪 det N).

* **.gralej file outputs:**
* **onea.gralej**

<'fido feeds the dog with biscuits'

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

s

{ :'npsg\_nprprule:fido'

npsg

{ :'lexicon:fido'

nprp

}

}

{ :'vpsgrule:feeds the dog with biscuits'

vpsg

{ :'lexicon:feeds'

vsg

}

{ :'np\_npsgrule:the dog with biscuits'

np

{ :'npsg\_det\_nsg\_pprule:the dog with biscuits'

npsg

{ :'lexicon:the'

det

}

{ :'lexicon:dog'

nsg

}

{ :'pprule:with biscuits'

pp

{ :'lexicon:with'

p

}

{ :'np\_npplrule:biscuits'

np

{ :'nppl\_nplrule:biscuits'

nppl

{ :'lexicon:biscuits'

npl

}

}

}

}

}

}

}

}

>

* **onec.gralej**

<'fido feeds the dog with biscuits'

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

s

{ :'np\_nprprule:fido'

np(

noun\_head: n(

noun\_num: sg,

prp: true))

{ :'lexicon:fido'

n(

noun\_num: sg,

prp: true)

}

}

{ :'vp\_sg\_or\_plrule:feeds the dog with biscuits'

vp(

verb\_head: v(

verb\_num: sg))

{ :'lexicon:feeds'

v(

verb\_num: sg)

}

{ :'np\_det\_sg\_n\_pprule:the dog with biscuits'

np(

noun\_head: n(

noun\_num: sg,

prp: false))

{ :'lexicon:the'

det

}

{ :'lexicon:dog'

n(

noun\_num: sg,

prp: false)

}

{ :'pprule:with biscuits'

pp

{ :'lexicon:with'

p

}

{ :'np\_nrule:biscuits'

np(

noun\_head: n(

noun\_num: pl,

prp: nprp))

{ :'lexicon:biscuits'

n(

noun\_num: pl,

prp: false)

}

}

}

}

}

}

>

* **twoc.gralej**

<'the student appeared to sleep'

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

s(

gap: n\_sem,

mood: indicative(

tense: tense),

vsem: 'mgsat(v\_sem)')

{ :'np\_rule:the student'

np(

nsem: $3 = n\_sem)

{ :'lexicon:the'

det

}

{ :'lexicon:student'

n(

nsem: student)

}

}

{ :'vp\_rule//2:appeared to sleep'

vp(

gap: $3,

mood: indicative(

tense: tense),

vsem: v\_sem(

agent: n\_sem\_or\_none,

ben: n\_sem\_or\_none,

exp: n\_sem\_or\_none,

gap\_type: type,

theme: n\_sem\_or\_none,

vform: past))

{ :'lexicon:appeared'

v(

vsem: appear(

agent: none,

ben: none,

exp: none,

gap\_type: none,

theme: $3,

vform: past))

}

{ :'inf\_torule:to sleep'

inf\_clause(

gap: n\_sem,

mood: infinitive,

rec: $3,

vsem: 'mgsat(v\_sem)')

{ :'lexicon:to'

toinf

}

{ :'lexicon:sleep'

v(

vsem: sleep(

agent: none,

ben: none,

exp: $3,

gap\_type: none,

theme: none,

vform: present))

}

}

}

}

>

* **Grammar Source Code:**
* **onea.pl**

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

s sub [].

npsg sub [].

vpsg sub [].

nppl sub [].

vppl sub [].

vsg sub [].

np sub [].

vpl sub [].

pp sub [].

p sub [].

nprp sub [].

det sub [].

nsg sub [].

npl sub [].

% rule

s\_sgrule rule

s

===>

cat> npsg,

cat> vpsg.

s\_plrule rule

s

===>

cat> nppl,

cat> vppl.

vpsgrule rule

vpsg

===>

cat> vsg,

cat> np.

vpplrule rule

vppl

===>

cat> vpl,

cat> np.

pprule rule

pp

===>

cat> p,

cat> np.

npsg\_nprprule rule

npsg

===>

cat> nprp.

npsg\_det\_nsgrule rule

npsg

===>

cat> det,

cat> nsg.

npsg\_det\_nsg\_pprule rule

npsg

===>

cat> det,

cat> nsg,

cat> pp.

nppl\_det\_nplrule rule

nppl

===>

cat> det,

cat> npl.

nppl\_det\_npl\_pprule rule

nppl

===>

cat> det,

cat> npl,

cat> pp.

nppl\_nplrule rule

nppl

===>

cat> npl.

nppl\_npl\_pprule rule

nppl

===>

cat> npl,

cat> pp.

np\_npsgrule rule

np

===>

cat> npsg.

np\_npplrule rule

np

===>

cat> nppl.

% lexicon

biscuits ---> npl.

dog ---> nsg.

fido ---> nprp.

feed ---> vpl.

feeds ---> vsg.

puppies ---> npl.

the ---> det.

with ---> p.

* **oneb.pl**

number sub [sg, pl].

nprp sub [true, false].

sg sub [].

pl sub [].

true sub [].

false sub [].

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

s sub [].

np sub [] intro [noun\_head : n].

n sub [] intro [noun\_num : number, prp : nprp].

vp sub [] intro [verb\_head : v].

v sub [] intro [verb\_num : number].

pp sub [].

p sub [].

det sub [].

% rule

s\_sg\_rule rule

s

===>

cat> (np, noun\_head :(noun\_num:sg)),

cat> (vp, verb\_head:(verb\_num:sg)).

s\_pl\_rule rule

s

===>

cat> (np, noun\_head :(noun\_num:pl)),

cat> (vp, verb\_head:(verb\_num:pl)).

vp\_sg\_or\_plrule rule

(vp, verb\_head:(verb\_num:sg))

===>

cat> (v, verb\_num:sg),

cat> np.

vp\_pl\_rule rule

(vp, verb\_head:(verb\_num:pl))

===>

cat> (v, verb\_num:pl),

cat> np.

pprule rule

pp

===>

cat> p,

cat> np.

np\_nrule rule

(np, noun\_head:(noun\_num:pl))

===>

cat> (n, noun\_num:pl).

np\_nprprule rule

(np, noun\_head:(noun\_num:sg, prp:true))

===>

cat> (n, noun\_num:sg, prp : true).

np\_det\_sg\_nrule rule

(np, noun\_head:(noun\_num:sg, prp:false))

===>

cat> det,

cat> (n, noun\_num:sg, prp : false).

np\_det\_pl\_nrule rule

(np, noun\_head:(noun\_num:pl, prp:false))

===>

cat> det,

cat> (n, noun\_num:pl, prp : false).

np\_det\_sg\_n\_pprule rule

(np, noun\_head:(noun\_num:sg, prp:false))

===>

cat> det,

cat> (n, noun\_num:sg, prp : false),

cat> pp.

np\_det\_pl\_n\_pprule rule

(np, noun\_head:(noun\_num:pl, prp:false))

===>

cat> det,

cat> (n, noun\_num:pl, prp : false),

cat> pp.

np\_npplrule rule

(np, noun\_head:(noun\_num:pl))

===>

cat> (n, noun\_num:pl),

cat> pp.

% lexicon

biscuits ---> (n, noun\_num:pl, prp:false).

dog ---> (n, noun\_num:sg, prp:false).

fido ---> (n, noun\_num:sg, prp:true).

feed ---> (v, verb\_num:pl).

feeds ---> (v, verb\_num:sg).

puppies ---> (n, noun\_num:pl, prp:false).

the ---> det.

with ---> p.

* **twob.pl**

:- 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].

toinf sub []. % infinitival to

n sub [].

v sub [].

p sub [].

det sub [].

% phrasal categories

cat sub [vproj,np].

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

s intro [mood:indicative].

inf\_clause intro [mood:infinitive, rec:n\_sem].

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 verbs

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

intro [vform:tense, agent:n\_sem\_or\_none, theme:n\_sem\_or\_none, ben:n\_sem\_or\_none, exp:n\_sem\_or\_none, gap\_type:type].

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

none sub [].

type sub [object, subject, none].

object sub [].

subject sub [].

% the following subtypes:

appear sub []

intro [vform:tense, agent:none, theme:n\_sem, ben:none, exp:none, gap\_type:none].

try sub []

intro [vform:tense, agent:n\_sem, theme:n\_sem, ben:none, exp:none, gap\_type:none].

promise sub []

intro [vform:tense, agent:n\_sem, theme:n\_sem, ben:n\_sem, exp:none, gap\_type:subject].

expect sub []

intro [vform:tense, agent:n\_sem, theme:n\_sem, ben:none, exp:none, gap\_type:object].

sleep sub []

intro [vform:tense, agent:none, theme:none, ben:none, exp:n\_sem, gap\_type:none].

% semantics for nouns

n\_sem sub [student, teacher].

student sub [].

teacher sub [].

%Rules

s\_rule rule

s

===>

cat> (np, nsem:Subj),

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

np\_rule rule

np

===>

cat> det,

cat> n.

% The student slept.

vp\_rule rule

(vp, vsem:(vform:past, exp:Subj),gap:Gap)

===>

cat> (v, vsem:(vform:past, theme:none, exp:Gap)).

% The student tried to sleep.

vp\_rule rule

(vp, vsem:(vform:past),gap:Gap)

===>

cat> (v, vsem:(vform:past, agent:Gap, theme:Gap, ben:none, exp:none)),

cat> (inf\_clause, rec:Gap).

% The student appeared to sleep.

vp\_rule rule

(vp, vsem:(vform:past),gap:Gap)

===>

cat> (v, vsem:(vform:past, agent:none, theme:Gap, ben:none, exp:none)),

cat> (inf\_clause, rec:Gap).

% The student expected the teacher to sleep.

vp\_rule rule

(vp, vsem:(vform:past),gap:Gap)

===>

cat> (v, vsem:(vform:past, agent:Gap, theme:Obj, ben:none, exp:none, gap\_type:object)),

cat> (np, nsem:Obj),

cat> (inf\_clause, rec:Obj).

% The student promised the teacher to sleep.

vp\_rule rule

(vp, vsem:(vform:past),gap:Gap)

===>

cat> (v, vsem:(vform:past, agent:Gap, theme:Obj, ben:Obj, exp:none, gap\_type:subject)),

cat> (np, nsem:Obj),

cat> (inf\_clause, rec:Gap).

inf\_torule rule

(inf\_clause, rec:Rec)

===>

cat> toinf,

cat> (v, vsem:(vform:present, exp:Rec)).

%Lexicon

appeared ---> (v, vsem:(appear, vform:past, agent:none, theme:Theme, ben:none, exp:none, gap\_type:none)).

expected ---> (v, vsem:(expect, vform:past, agent:Agent, theme:Theme, ben:none, exp:none, gap\_type:object)).

promised ---> (v, vsem:(promise, vform:past, agent:Agent, theme:Theme, ben:Ben, exp:none, gap\_type:subject)).

sleep ---> (v, vsem:(sleep, vform:present, agent:none, theme:none, ben:none, exp:Exp, gap\_type:none)).

slept ---> (v, vsem:(sleep, vform:past, agent:none, theme:none, ben:none, exp:Exp, gap\_type:none)).

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

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

the ---> det.

to ---> toinf.

tried ---> (v, vsem:(try, vform:past, agent:Agent, theme:Theme, ben:none, exp:none, gap\_type:none)).