Name: Yuhan Shao
Student Number: 1002281327
UTORid: shaoyuha

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)).
```