LAB3B REPORT

May 7, 2019

Wenchang Liu

ID: 10406141

School of Computer Science

Contents

1 EX1 BUILDING A CFG

1. Steel is an alloy.

   S -> NN VP .

   NP -> NN .

   VP -> VBZ NP.

   NP -> DT NN .

   NN -> Steel .

   VBZ -> is .

   DT -> an .

   NN -> alloy .

2. Steel contains carbon.

   S -> NN VP .

   NP -> NN .

   VP -> VBZ NP .

   NN -> Steel .

   VBZ -> contains .

   NN -> carbon .

3. We can combine them together.

   S -> NN VP .

   NP -> NN .

   VP -> VBZ NP .

   NP -> DT NN .

   NN -> Steel . NN -> alloy . NN -> carbon .

   VBZ -> is . VBZ -> contains.

   DT -> an .

2 Ex2 Building a CFG in Prolog

1. Translate the CFG grammar from the previous exercise into Prolog using difference lists.

   See the prolog file *cfg.pl* in appendix A.

2. Check if the grammar is able to recognise the two sentences.

   Yes.

3. Generate all the sentences which can be recognised by this grammar.

   Query: s(X, []).

   X = [steel, is, steel] ;

   X = [steel, is, alloy] ;

   X = [steel, is, carbon] ;

   X = [steel, is, an, steel] ;

   X = [steel, is, an, alloy] ;

   X = [steel, is, an, carbon] ;

   X = [steel, contains, steel] ;

   X = [steel, contains, alloy] ;

   X = [steel, contains, carbon] ;

   X = [steel, contains, an, steel] ;

   X = [steel, contains, an, alloy] ;

   X = [steel, contains, an, carbon] ;

   X = [alloy, is, steel] ;

   X = [alloy, is, alloy] ;

   X = [alloy, is, carbon] ;

   X = [alloy, is, an, steel] ;

   X = [alloy, is, an, alloy] ;

   X = [alloy, is, an, carbon] ;

   X = [alloy, contains, steel] ;

X = [alloy, contains, alloy] ;

X = [alloy, contains, carbon] ;

X = [alloy, contains, an, steel] ;

X = [alloy, contains, an, alloy] ;

X = [alloy, contains, an, carbon] ;

X = [carbon, is, steel] ;

X = [carbon, is, alloy] ;

X = [carbon, is, carbon] ;

X = [carbon, is, an, steel] ;

X = [carbon, is, an, alloy] ;

X = [carbon, is, an, carbon] ;

X = [carbon, contains, steel] ;

X = [carbon, contains, alloy] ;

X = [carbon, contains, carbon] ;

X = [carbon, contains, an, steel] ;

X = [carbon, contains, an, alloy] ;

X = [carbon, contains, an, carbon].

3 Ex3 DCGs

1. Rephrase the previous grammar into a DCG.

   See prolog files *dcg.pl* in appendix B.

2. Check if you can recognise and generate the same sentences.

   Yes. It gets the same results as CFG.

4 Ex4 CCGs

1. Create a CCG Grammar which recognises these sentences.

   Steel: N

is: $(S\backslash N)/NP$

an: NP/N

alloy: N

contains: $(S\backslash N)/N$

carbon: N

Does: $S/(S\backslash NP)/NP$

Ferrite: NP

have: $(S\backslash NP)/N$

high: N/N

hardness: N

Which: $S/(S\backslash NP)/N$

material: N

has: $(S\backslash NP)/NP$

the: NP/N

lowest: N/N

tensile: N/N

strength: N

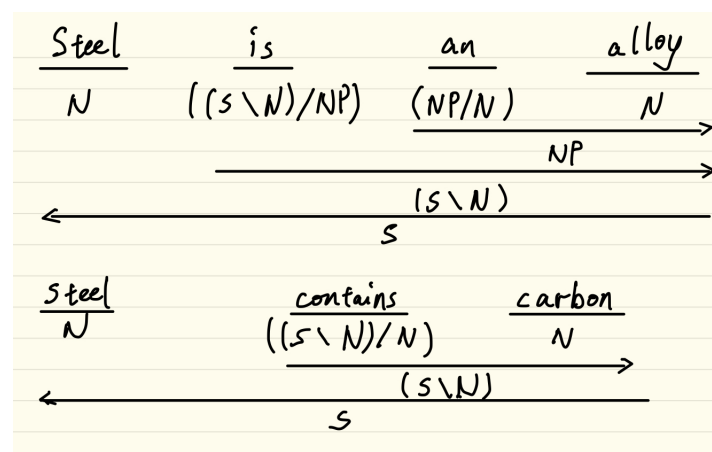2. Draw the derivation trees for each sentence.

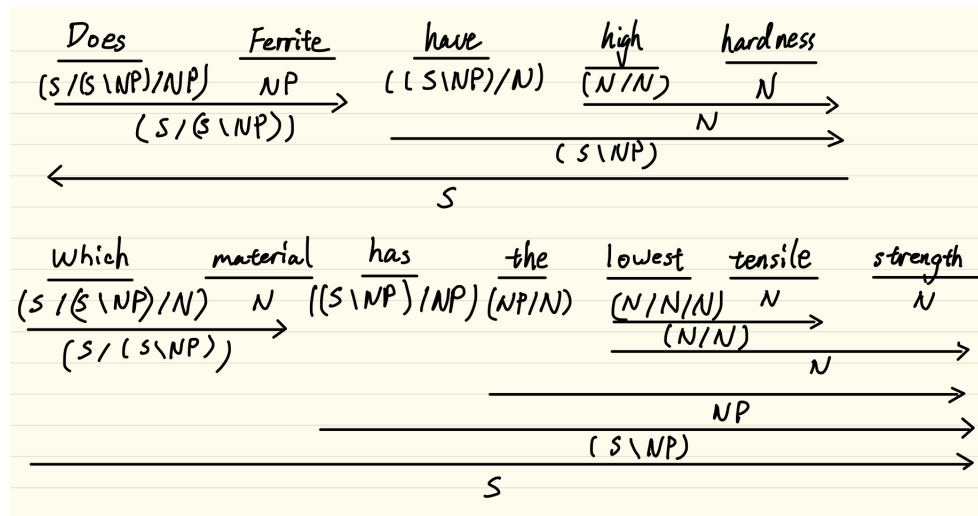**Figure 1:** derivation trees for statements

Figure 2: derivation trees for questions

5   Ex5 CCG Syntactic Parser

1. Implement a syntactic parser in NLTK for the grammar that you designed in the previous exercise.

   See the python file in appendix C.

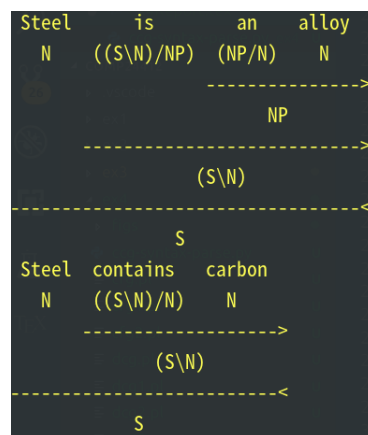2. Print the derivation tree for the four sentences.

Figure 3: CCG syntactic parse for statements

```
Does     Ferrite     have     high    hardness
(NP/NP)    NP      ((S\NP)/N)  (N/N)     N
----------------->
      NP
                         ----------------->
                                N
                         ----------------------------->
                              (S\NP)
----------------------------------------------<
                    S
  Which        material      has      the    lowest  tensile  strength
((S/(S\NP))/N)     N      ((S\NP)/NP)  (NP/N)  (N/N)   (N/N)      N
------------------------>
      (S/(S\NP))
                                              ------------------->
                                                       N
                                          ----------------------------->
                                                     N
                                      ------------------------------------->
                                                  NP
                              ------------------------------------------------->
                                              (S\NP)
----------------------------------------------------------------------------->
                                   S
```
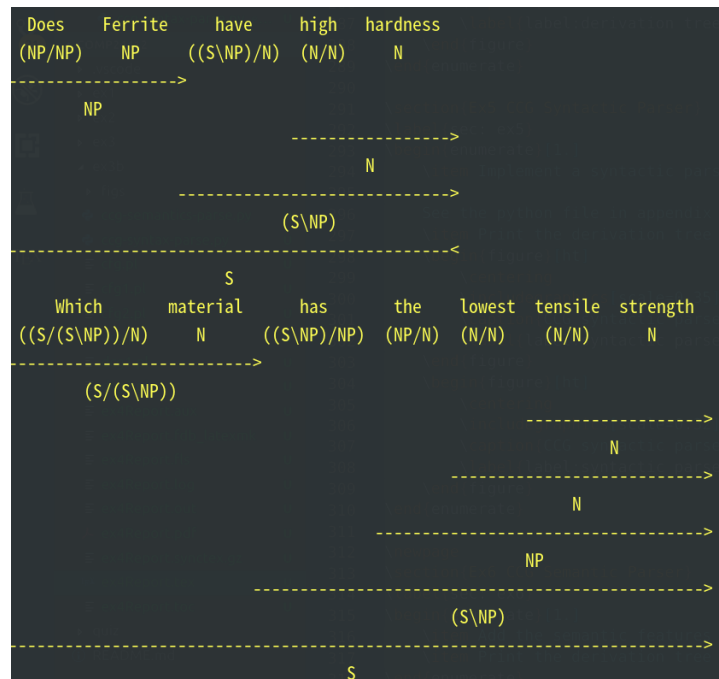
Figure 4: CCG syntactic parse for questions

6   Ex6 CCG Semantic Parser

1. Add the semantic features into your grammar.

   See the python file with semantic features in appendix D.

2. Print the derivation tree for the sentences in the corpus using the semantic features.
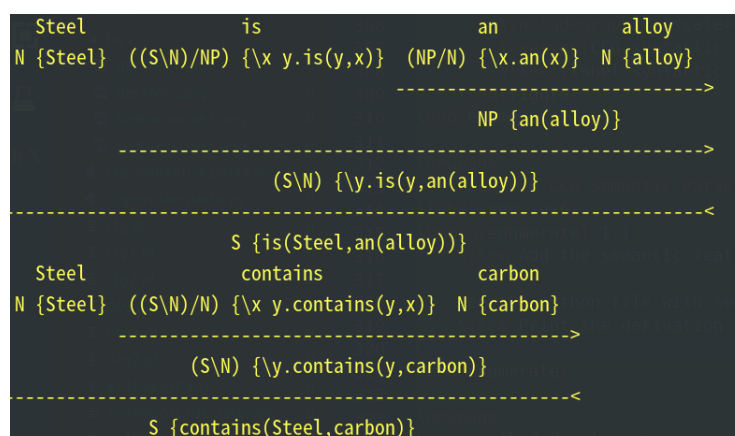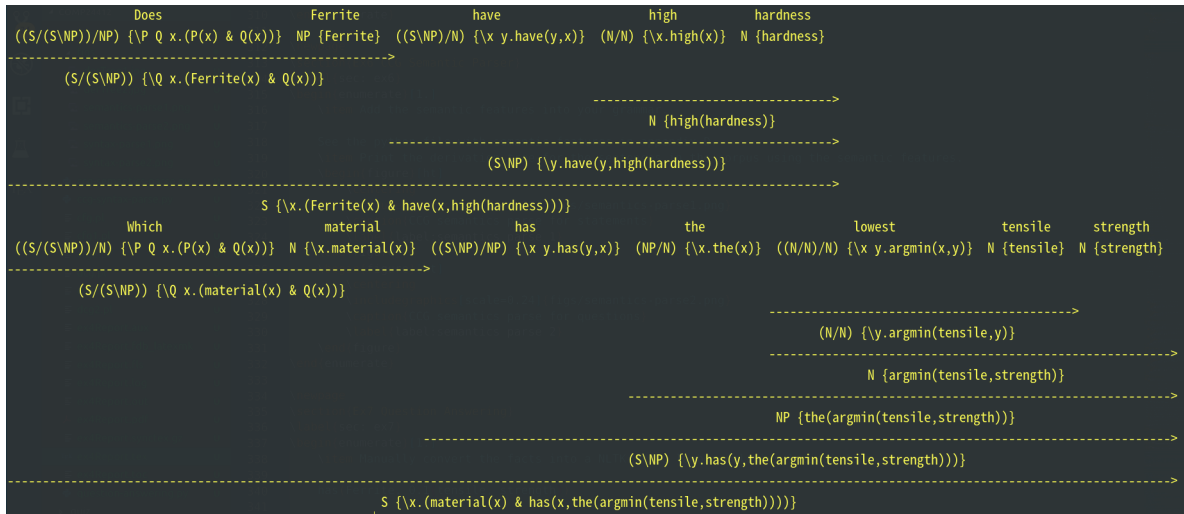
```
  Steel              is               an          alloy
N {Steel}  ((S\N)/NP) {\x y.is(y,x)}  (NP/N) {\x.an(x)}  N {alloy}
                                      ----------------------------->
                                             NP {an(alloy)}
            ----------------------------------------------------->
                      (S\N) {\y.is(y,an(alloy))}
----------------------------------------------------------------<
                  S {is(Steel,an(alloy))}
  Steel               contains              carbon
N {Steel}  ((S\N)/N) {\x y.contains(y,x)}  N {carbon}
            ------------------------------------------->
                 (S\N) {\y.contains(y,carbon)}
----------------------------------------------------<
          S {contains(Steel,carbon)}
```

Figure 5: CCG semantics parse for statements

Figure 6: CCG semantics parse for questions

7   Ex7 Question Answering

1. Manually convert the facts into a NLTK-style lambda-calculus logical form.

   has(Ferrite, 0.1(tensile, strength))

   has(Ferrite, low(hardness))

   has(Perlite, 0.3(tensile, strength))

   has(Perlite, medium(hardness))

   has(Austenite, 0.4(tensile, strength))

   has(Austenite, medium(hardness))

   has(Cementite, 0.7(tensile, strength))

   has(Cementite, high(hardness))

2. For the two questions, adapt the semantic features of your previous grammar to match the predicate form in the KB.

   I slightly adjust the semantics of the previous exercise so I can do the matchings:

   Match Ferrite(x) to Ferrite(Ferrite)

   Match has(x, high(hardness)) to has(Cementite, high(hardness))

   Match argmin(tensile, strength)) to 0.1(tensile, strength)

   Match has(x, argmin(tensile, strength)) to has(Ferrite, 0.1(tensile, strength))

3. Run the parser and print the parse trees for the two questions.

Parser output:

$S\{\backslash x.(Ferrite(x)\&has(x,high(hardness)))\}$

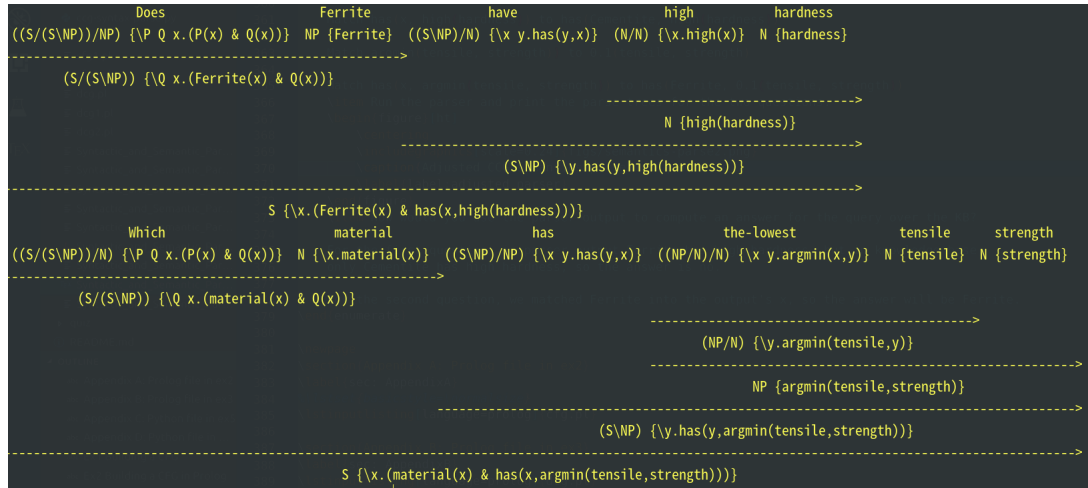$S\{\backslash x.(material(x)\&has(x,argmin(tensile,strength)))\}$

Figure 7: Adjusted CCG semantics parse for questions

4. How would you use the parser output to compute an answer for the query over the KB?

For the first question, in parser output x has high hardness, and in knowledge base, only Cementite has high hardness, so we match x as Cementite, which is not agree with Ferrite(x), so the answer is no.

For the second question, we firstly search for argmin in the knowledge base, which is 0.1 among 0.1, 0.3, 0.4 and 0.7, and then we can match the "has" predicate, and here x is Ferrite, also, Ferrite is a kind of material, so material(Ferrite) is true, so the answer of the query is Ferrite.

8 APPENDIX A: PROLOG FILE FOR EX2

```
s(X, Z) :- nn(X, Y), vp(Y, Z).
np(X, Z) :- nn(X, Z).
np(X, Z) :- dt(X, Y), nn(Y, Z).
vp(X, Z) :- vbz(X, Y), np(Y, Z).


nn([steel|W], W).
nn([alloy|W], W).
nn([carbon|W], W).
vbz([is|W], W).
vbz([contains|W], W).
dt([an|W], W).
```

9 APPENDIX B: PROLOG FILE FOR EX3

```
s --> nn, vp.
np --> nn.
np --> dt, nn.
vp --> vbz, np.


nn --> [steel].
nn --> [alloy].
nn --> [carbon].
vbz --> [is].
vbz --> [contains].
dt --> [an].
```

10 APPENDIX C: PYTHON FILE FOR EX5

```
from nltk.ccg import chart, lexicon

lex = lexicon.fromstring('''
    :- S, N, NP
    Steel => N
    is => (S\\N)/NP
```

```
        an => NP/N
        alloy => N
        contains => (S\\N)/N
        carbon => N
        Does => NP/NP
        Ferrite => NP
        have => (S\\NP)/N
        high => N/N
        hardness => N
        Which => S/(S\\NP)/N
        material => N
        has => (S\\NP)/NP
        the => NP/N
        lowest => N/N
        tensile => N/N
        strength => N
        ''',
        False)


parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
for parse in parser.parse("Steel_is_an_alloy".split()):
    chart.printCCGDerivation(parse)
    break


for parse in parser.parse("Steel_contains_carbon".split()):
    chart.printCCGDerivation(parse)
    break


for parse in parser.parse(
    "Does_Ferrite_have_high_hardness".split()):
    chart.printCCGDerivation(parse)
    break


for parse in parser.parse(
    "Which_material_has_the_lowest_tensile_strength".split()):
```

```
chart.printCCGDerivation(parse)
break
```

11 APPENDIX D: PYTHON FILE FOR ex6

```
from nltk.ccg import chart, lexicon

lex = lexicon.fromstring('''
    :- S, N, NP
    Steel => N {Steel}
    is => (S\\N)/NP {\\x y.is(y, x)}
    an => NP/N {\\P exist x.P(x)}
    alloy => N {alloy}
    contains => (S\\N)/N {\\x y.contains(y, x)}
    carbon => N {carbon}
    Does => S/(S\\NP)/NP {\\P Q x.(P(x) & Q(x))}
    Ferrite => NP {Ferrite}
    have => (S\\NP)/N {\\x y.have(y, x)}
    high => N/N {\\x.high(x)}
    hardness => N {hardness}
    Which => S/(S\\NP)/N {\\P Q x.(P(x) & Q(x))}
    material => N {\\x.material(x)}
    has => (S\\NP)/NP {\\x y.has(y, x)}
    the => NP/N {\\x. the(x)}
    lowest => N/N/N {\\x y.argmin(x, y)}
    tensile => N {tensile}
    strength => N {strength}
    ''',
    True)


parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
for parse in parser.parse("Steel is an alloy".split()):
    chart.printCCGDerivation(parse)
    break
```

```
for  parse  in  parser.parse("Steel_contains_carbon".split()):
    chart.printCCGDerivation(parse)
    break


for  parse  in  parser.parse(
    "Does_Ferrite_have_high_hardness".split()):
    chart.printCCGDerivation(parse)
    break


for  parse  in  parser.parse(
    "Which_material_has_the_lowest_tensile_strength".split()):
    chart.printCCGDerivation(parse)
    break
```

12 APPENDIX E: PYTHON FILE FOR EX7

```
from  nltk.ccg  import  chart,  lexicon

lex = lexicon.fromstring('''
    :- S, N, NP
    Does => S/(S\\NP)/NP {\\P Q x.(P(x) & Q(x))}
    Ferrite => NP {Ferrite}
    have => (S\\NP)/N {\\x y.has(y, x)}
    high => N/N {\\x.high(x)}
    hardness => N {hardness}
    Which => S/(S\\NP)/N {\\P Q x.(P(x) & Q(x))}
    material => N {\\x.material(x)}
    has => (S\\NP)/NP {\\x y.has(y, x)}
    the-lowest => NP/N/N {\\x y.argmin(x, y)}
    tensile => N {tensile}
    strength => N {strength}
    ''',
    True)


parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
```

```python
for parse in parser.parse(
    "Does Ferrite have high hardness".split()):
    chart.printCCGDerivation(parse)
    break


for parse in parser.parse(
    "Which material has the-lowest tensile strength".split()):
    chart.printCCGDerivation(parse)
    break
```