LAB3B REPORT

May 5, 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\N)/NP

an: NP/N

alloy: N

contains: (S\N)/N

carbon: N

Does: S/(S\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

2. Draw the derivation trees for each sentence.

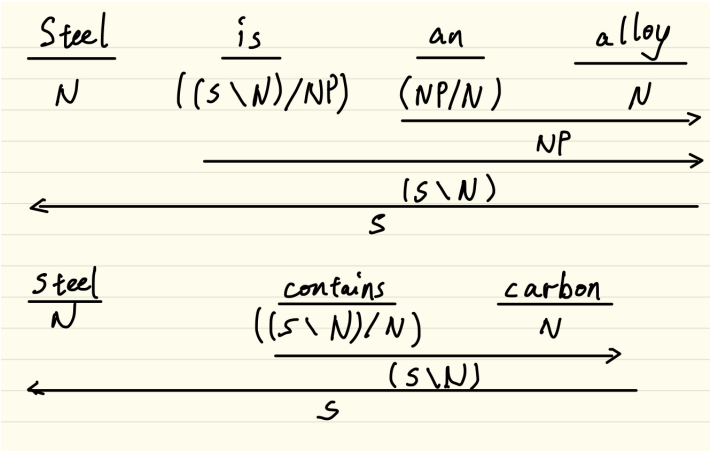

Figure 1: derivation trees for statements

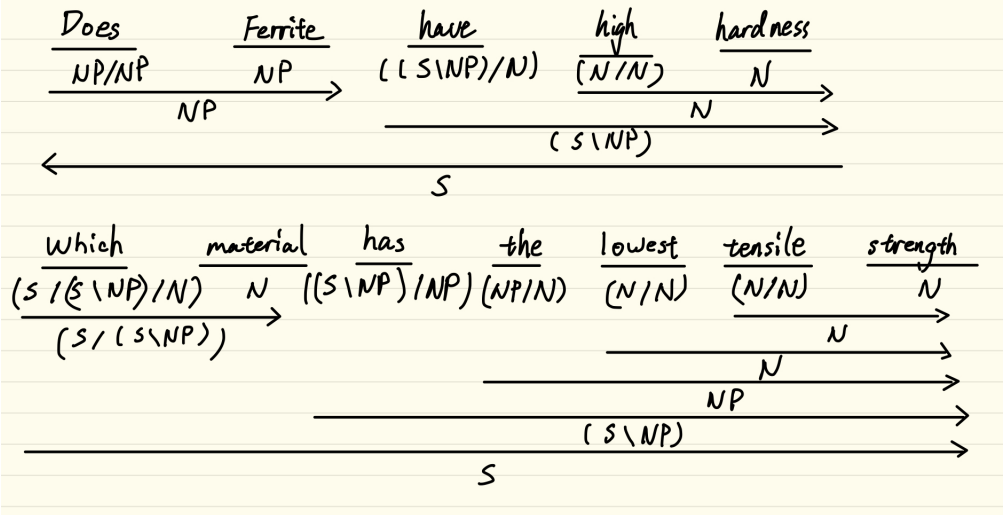

Figure 2: derivation trees for questions

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.

```
Steel      is        an     alloy
  N    ((S\N)/NP)  (NP/N)     N
                  --------------->
                       NP
       --------------------------->
                  (S\N)
<---------------------------------
                  S
Steel  contains    carbon
  N    ((S\N)/N)     N
       ------------------->
              (S\N)
<-------------------------
              S
```

Figure 3: CCG syntactic parse for statements

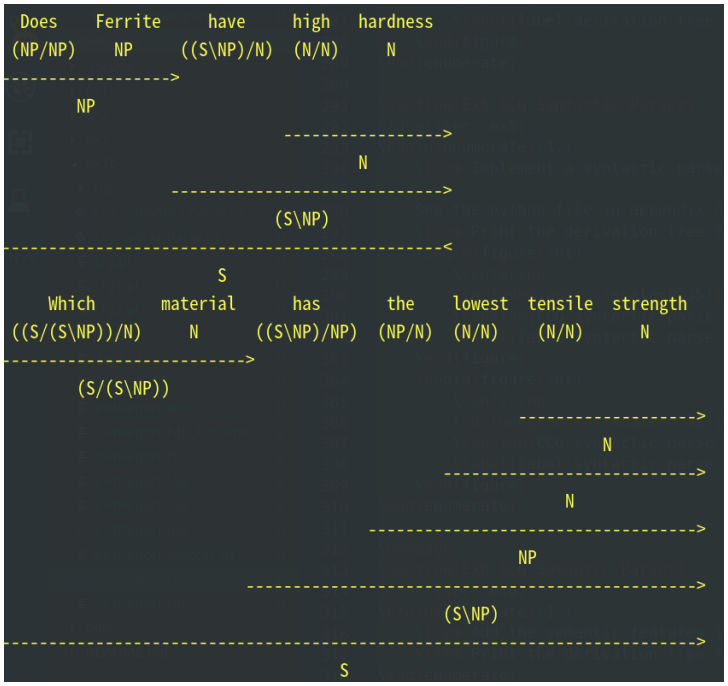

Figure 4: CCG syntactic parse for questions

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.

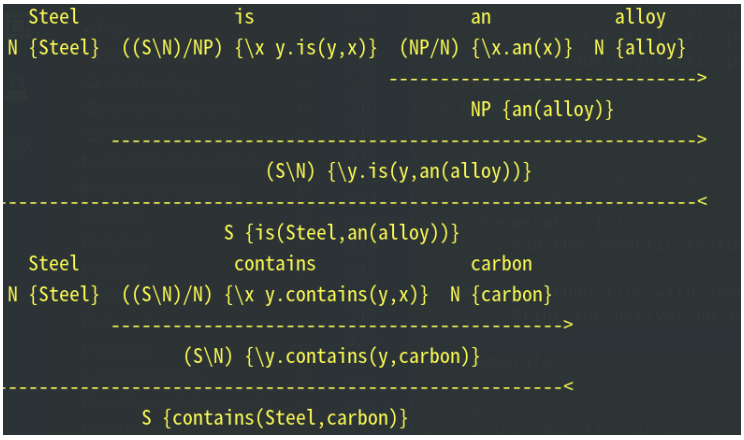

Figure 5: CCG semantics parse for statements

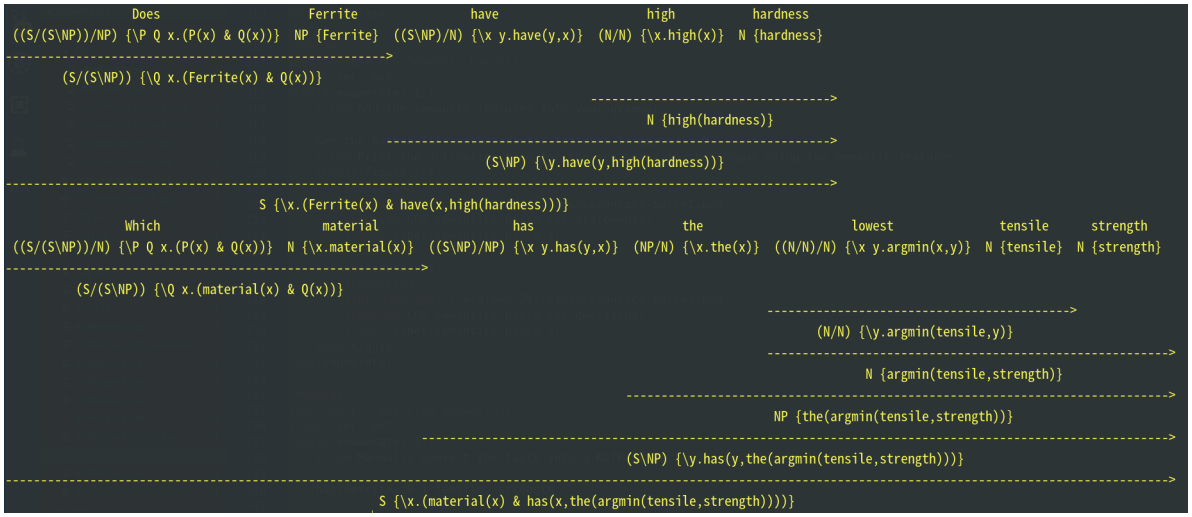

Figure 6: CCG semantics parse for questions

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.

   Match Ferrite(x) to Ferrite(Ferrite)

   Match have(x, high(hardness)) to has(Ferrite, low(hardness))

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

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

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

See figure CCG semantics parse for questions in the previous exercise.

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

   For the first question, in output Ferrite has high hardness, but in knowledge base, Ferrite has low hardness, so the answer is no.

   For the second question, we matched Ferrite into the output's x, so the answer will be Ferrite.

8 Appendix A: Prolog file in 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 in 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 in 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 in 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 {\\x.an(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
```