CS4248 Assignment 3

Heng Low Wee (U096901R)

1. (Refer to algorithm at the end)

2. Input: "love stops"

Chart[0]:

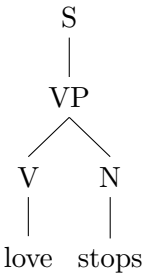
✓	S0	$\gamma \rightarrow \bullet$ S	[0,0]	Dummy start state
✓	S1	S $\rightarrow \bullet$ N VP	[0,0]	Predictor
✓	S2	S $\rightarrow \bullet$ VP	[0,0]	Predictor
✓	S3	VP $\rightarrow \bullet$ V	[0,0]	Predictor
✓	S4	VP $\rightarrow \bullet$ V N	[0,0]	Predictor

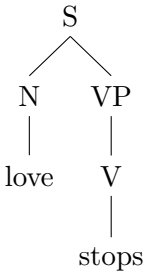
Chart[1]:

✓	S5	N $\rightarrow$ love $\bullet$	[0,1]	Scanner
✓	S6	V $\rightarrow$ love $\bullet$	[0,1]	Scanner
✓	S7	S $\rightarrow$ N $\bullet$ VP	[0,1]	Completer (S5)
✓	S8	VP $\rightarrow$ V $\bullet$	[0,1]	Completer (S6)
✓	S9	VP $\rightarrow$ V $\bullet$ N	[0,1]	Completer (S6)
✓	S10	VP $\rightarrow \bullet$ V	[1,1]	Predictor
✓	S11	VP $\rightarrow \bullet$ V N	[1,1]	Predictor
✓	S12	S $\rightarrow$ VP $\bullet$	[1,1]	Completer (S8)
✓	S13	$\gamma \rightarrow$ S $\bullet$	[1,1]	Completer (S12)

Chart[2]:

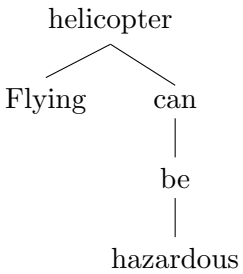
✓	S14	N $\rightarrow$ stops $\bullet$	[1,2]	Scanner
✓	S15	V $\rightarrow$ stops $\bullet$	[1,2]	Scanner
✓	S16	VP $\rightarrow$ V N $\bullet$	[1,2]	Completer (S14)
✓	S17	VP $\rightarrow$ V $\bullet$	[1,2]	Completer (S15)
✓	S18	VP $\rightarrow$ V $\bullet$ N	[1,2]	Completer (S15)
✓	S19	S $\rightarrow$ VP $\bullet$	[1,2]	Completer (S16)
✓	S20	S $\rightarrow$ N VP $\bullet$	[1,2]	Completer (S16)
✓	S21	$\gamma \rightarrow$ S $\bullet$	[1,2]	Completer (S19)
✓	S22	$\gamma \rightarrow$ S $\bullet$	[1,2]	Completer (S20)



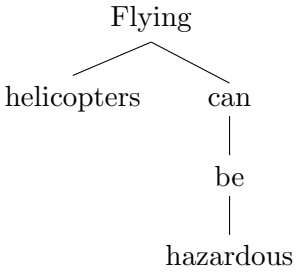


3. The conventional CKY algorithm has time complexity of $O(n^3)$. To go through each item in the table takes n^2, and for each item checking through all the sub-trees takes n. Now that we consider probabilistic CKY, for each sub-tree there are multiple ways of parsing them, each with its own probability value. If we consider the worst case, there is an additional $|G|$ steps, where G is the grammar. Thus, time complexity is $O(|G| \cdot n^3)$.

4. Sentence in consideration: "Flying helicopters can be hazardous."



In this interpretation, "Flying" modifies "helicopter". It describes a third-person perspective and means helicopters that are flying are hazardous.



In this interpretation, "Flying" is a verb. It describes a first-person perspective and means the act of piloting helicopters can be hazardous

Algorithm 1 convertRegexToCFG(R, G_t, A_{prev}) return G

R: Regex
G_t: Grammar
A_{prev}: Previous symbol or S (First call)
if R is terminal symbol **then**
  G_t.add($A_{prev} \rightarrow R$)
  return G_t
else if R in form of (r) **then**
  Generate new non-terminal symbol, A_{new}
  G_t.add($A_{prev} \rightarrow A_{new}$)
  return convertRegexToCFG(r, G_t, A_{new})
else if R in form of $r+$ **then**
  return convertRegexToCFG($rr*$, G_t, A_{new})
else if R in form of $r*$ **then**
  Generate new non-terminal symbol, A_{new}
  G_t.add($A_{prev} \rightarrow A_{prev}A_{new}$)
  G_t.add($A_{prev} \rightarrow \epsilon$)
  return convertRegexToCFG(r, G_t, A_{new})
else if R in form of $r?$ **then**
  Generate new non-terminal symbol, A_{new}
  G_t.add($A_{prev} \rightarrow A_{new}$)
  G_t.add($A_{prev} \rightarrow \epsilon$)
  return convertRegexToCFG(r, G_t, A_{new})
else if R in form of $r\{d\}$ **then**
  Generate new non-terminal symbol, A_{new}
  $A_{temp} \leftarrow null$
  for 1 to d **do**
    A_{temp}.append(A_{new})
  end for
  G_t.add($A_{prev} \rightarrow A_{temp}$)
  return convertRegexToCFG(r, G_t, A_{new})
else if R in form of $r\{d,\}$ **then**
  Generate new non-terminal symbol, A_{new}
  $A_{temp} \leftarrow null$
  for 1 to d **do**
    A_{temp}.append(A_{new})
  end for
  G_t.add($A_{prev} \rightarrow A_{temp}$)
  return convertRegexToCFG($r*$, G_t, A_{new})
else if R in form of $r\{d_1,d_2\}$ **then**
  Generate new non-terminal symbol, A_{new}
  for d_i in d_1 to d_2 **do**
    convertRegexToCFG($r\{d_i\}$, G_t, A_{new})
  end for
else if R in form of $r_1 \mid r_2$ **then**
  Generate 2 new non-terminal symbols, A and B
  G_t.add($A_{prev} \rightarrow A$)
  G_t.add($A_{prev} \rightarrow B$)
  return convertRegexToCFG(r_2, convertRegexToCFG(r_1, G_t, A), B)
else if R in form of r_1r_2 **then**
  Generate 2 new non-terminal symbols, A and B
  G_t.add($A_{prev} \rightarrow A$)
  G_t.add($A_{prev} \rightarrow B$)
  return convertRegexToCFG(r_2, convertRegexToCFG(r_1, G_t, A), B)
end if