

# تحلیل الگوریتم cyk

محمد مهدی حیدری  
۹۴۳۱۳۰۶

۱۶ خرداد ۱۳۹۸

در فایل `cyk_main.py` چند تابع کمکی برای خواندن ورودی و گرامر از فایل متنی نوشته شده. در انتهای هر فایل باید یک خط خالی وجود داشته باشد. سمت راست و چپ هر قانون گرامر باید با  $\rightarrow$  جدا شده باشد و نتایج آن می‌تواند با  $|$ ،  $or$  شود. یک تابع نیز برای نمایش نتیجه الگوریتم نوشته شده که در ابتدای هر خط طول زیرمجموعه‌های مورد بررسی از ورودی نوشته شده است. نتیجه‌ی موجود در هر خانه با  $\{ \}$  مشخص شده و اگر خالی باشد به جای آن - گذاشته شده. در تابع مربوط به الگوریتم ابتدا ترمینال‌های موجود در ورودی در نظر گرفته می‌شوند. قواعدی که منجر به تولید ترمینال می‌شوند به طور جداگانه ذخیره شده‌اند و فقط در مرحله اول مورد نیاز هستند. برای هر ترمینال در ورودی متغیری در را پیدا می‌کنیم که در سمت چپ یک قاعده که آن ترمینال را تولید می‌کند، حضور داشته باشد. در مرحله بعد زیر رشته‌های دوتایی را بررسی می‌کنیم. برای رخ دادن هر کدام باید در سمت راست قواعدی که دو متغیر تولید می‌کنند به دنبال آنها بگردیم و متغیر سمت چپ را ذخیره کنیم. در مراحل بعدی برای هر زیررشته ورودی چک می‌کنیم که تحت چه حالت‌هایی قابل ساخته شدن هستند. مثلاً  $aba$  از ترکیب  $a + ab$  یا  $a + ba$  تولید می‌شود که نتایج هر یک در جدول موجود است و با یک تابع کمکی تمام حالت‌های ممکن را می‌سازیم و باز به دنبال حالت‌های رخداد آن می‌گردیم در مرحله آخر اگر متغیر آغازی در نتیجه وجود داشته باشد یعنی می‌توانیم با شروع از آن و استفاده از قواعد موجود در گرامر، رشته داده شده را بسازیم.

```

1 import os
2
3
4 def create_cell(first, second):
5     """
6     creates set of string from concatenation of each character in first
7     to each character in second
8     :param first: first set of characters
9     :param second: second set of characters
10    :return: set of desired values
11    """
12    res = set()
13    if first == set() or second == set():
14        return set()
15    for f in first:
16        for s in second:
17            res.add(f+s)
18    return res
19
20
21 def read_grammar(filename="./grammar.txt"):
22     """
23     reads the rules of a context free grammar from a text file
24     :param filename: name of the text file in current directory
25     :return: two lists. v_rules lead to variables and t_rules
26     lead to terminals.
27     """
28    filename = os.path.join(os.getcwd(), filename)
29    with open(filename) as grammar:
30        rules = grammar.readlines()
31        v_rules = []
32        t_rules = []
33
34        for rule in rules:
35            left, right = rule.split(" -> ")
36
37            # for two or more results from a variable
38            right = right[:-1].split(" | ")
39            for ri in right:
40
41                # it is a terminal
42                if str.islower(ri):
43                    t_rules.append([left, ri])
44
45                # it is a variable
46                else:
47                    v_rules.append([left, ri])
48    return v_rules, t_rules
49
50
51 def read_input(filename="./input.txt"):
52     """
53     reads the inputs from a text file
54     :param filename: name of the text file in current directory
55     :return: list of inputs
56     """
57    filename = os.path.join(os.getcwd(), filename)
58    res = []
59    with open(filename) as inp:
60        inputs = inp.readlines()
61        for i in inputs:
62
63            # remove \n
64            res.append(i[:-1])
65    return res
66
67
68 def cyk_alg(varies, terms, inp):
69     """
70     implementation of CYK algorithm
71     :param varies: rules related to variables
72     :param terms: rules related to terminals
73     :param inp: input string

```

```

74 :return: resulting table
75 """
76
77 length = len(inp)
78 var0 = [va[0] for va in varies]
79 var1 = [va[1] for va in varies]
80
81 # table on which we run the algorithm
82 table = [[set() for _ in range(length-i)] for i in range(length)]
83
84 # Deal with variables
85 for i in range(length):
86     for te in terms:
87         if inp[i] == te[1]:
88             table[0][i].add(te[0])
89
90 # Deal with terminals
91 # its complexity is O(|G|*n^3)
92 for i in range(1, length):
93     for j in range(length - i):
94         for k in range(i):
95             row = create_cell(table[k][j], table[i-k-1][j+k+1])
96             for ro in row:
97                 if ro in var1:
98                     table[i][j].add(var0[var1.index(ro)])
99
100 # if the last element of table contains "S" the input belongs to the grammar
101 return table
102
103
104 def show_result(tab, inp):
105     """
106     this function prints the procedure of cyk.
107     in the end there is a message showing if the input
108     belongs to the grammar
109     :param tab: table
110     :param inp: input
111     :return: None
112     """
113     for c in inp:
114         print("\t{}".format(c), end="\t")
115     print()
116     for i in range(len(inp)):
117         print(i+1, end=" ")
118         for c in tab[i]:
119             if c == set():
120                 print("\t{}".format("_"), end="\t")
121             else:
122                 print("\t{}".format(c), end=" ")
123         print()
124
125     if 'S' in tab[len(inp)-1][0]:
126         print("The input belongs to this context free grammar!")
127     else:
128         print("The input does not belong to this context free grammar!")
129
130
131 if __name__ == '__main__':
132     v, t = read_grammar()
133     r = read_input()[0]
134     ta = cyk_alg(v, t, r)
135     show_result(ta, r)

```

برای مشاهده کد می‌توانید به اینجا مراجعه کنید.