

LR(0)语法分析器的实现代码 (python)

 kimotoo

关注

2019.01.18 18:04:40 字数 1,279 阅读 1,119

- 构造LR(0)项目集：
 - 构造I的闭包CLOSURE(I)的算法如下：
 - I的任何项目都属于CLOSURE(I)；
 - 若 $A \rightarrow \alpha \cdot B \beta$ 属于CLOSURE(I)，对任何产生式 $B \rightarrow \gamma$ ， $B \rightarrow \cdot \gamma$ 也属于CLOSURE(I)；
 - 重复执行上述两步骤直至CLOSURE(I)不再增大为止。
 - 实现代码如下


```
1 def get_CLOSURE(tmp):
2     # 生成闭包
3     CLOSURE = []
4     for it in tmp:
5         if(it not in CLOSURE):
6             CLOSURE.append(it)
7         x, y = it.split(".")
8         if(y == ""):
9             continue
10        v = y[0]
11        if(v in VN):
12            res = get_VN_gram(v) # 返回非终结符产生的A->.aBb形式
13            for re in res:
14                if(re not in CLOSURE):
15                    CLOSURE.append(re)
16    return CLOSURE
```

- Go(I,a)函数构造算法
 - I为当前状态，X为文法符号，J为I中所有形如 $A \rightarrow \alpha \cdot X \beta$ 的项目的后续项目所组成的集合，而CLOSURE(J)就是项目集I关于X的后续状态
 - 实现代码如下

```
1 def go(item, v):
2     #生成并返回下一个item
3     tmp = []
4     for it in item:
5         x, y = it.split(".")
6         if(y!=""):
7             if(y[0] == v):
8                 new_it = x + y[0] + "." + y[1:]
9                 tmp.append(new_it)
10    if(len(tmp)!=0):
11        new_item = get_CLOSURE(tmp)
12        #print(tmp)
13        #print("go(item, "+v+" ) = " + str(new_item))
14    return new_item
```

- 判别LR项目集是否合法：
 - 无移进项目和规约项目并存
 - 无多个规约项目并存
 - 代码如下：



 kimotoo

关注

总资产1 (约0.11元)

简单赋值语句翻译(python)

阅读 109

LR(0)语法分析器的实现代码 (python)

阅读 1,118

- 推荐阅读

好人没好报，讨好型人格究竟有多危险？

阅读 6,044
- Visual Studio Code 远程开发探秘

阅读 11,356
- 知乎热门一句话打脸父母，揭开多少人不敢说的隐痛

阅读 29,979
- 59岁费翔近照曝光：你妈妈心中的梦中情人，真是帅了30年

阅读 13,596
- 杨烁还不是最油的

阅读 6,675



```

6     for item in items:
7         for it in item:
8             x, y = it.split(".")
9             if(y == ""):
10                if(has_protocol != 0 or has_shift != 0):
11                    return False
12                has_protocol = 1
13            else:
14                if(y[0] in VT):
15                    has_shift = 1
16        return True

```

• 构造LR(0)分析表

• 构造算法：

1. 假定项目集规范族 $C = \{I_0, I_1, \dots, I_n\}$ 。令每一个项目集 I_k 的下标 k 作为分析器的状态。分析表的ACTION子表和GOTO子表可按如下方法构造
2. 令那个包含项目 $S' \rightarrow \cdot S$ 的集合 I_k 的下标 k 为分析器的初态。
3. 若项目 $A \rightarrow \alpha \cdot a\beta$ 属于 I_k 且 $GO(I_k, a) = I_j$, a 为终结符，置 $ACTION[k, a]$ 为“把 (j, a) 移进栈”，简记为“sj”。
4. 若项目 $A \rightarrow \alpha \cdot$ 属于 I_k ，对任何终结符 a (或结束符 #)，置 $ACTION[k, a]$ 为“用产生式 $A \rightarrow \alpha$ 进行归约”，简记为“rj” (假定产生式 $A \rightarrow \alpha$ 是文法 G 的第 j 个产生式)。
5. 若项目 $S' \rightarrow S \cdot$ 属于 I_k ，则置 $ACTION[k, \#]$ 为“接受”，简记为“acc”。
6. 若 $GO(I_k, A) = I_j$ ， A 为非终结符，则置 $GOTO[k, A] = j$ 。
7. 分析表中凡不能用规则1至4填入信息的空白格均填上“err”。

```

1  def get_lr_table():
2      #构建lr分析表
3      init_lr_table()
4      lr_is_legal()
5      i=0
6      j=0
7      for item in items:
8          for it in item:
9              x, y = it.split(".")
10             if(y==""): #判断是否写入ACTION
11                 if (it == "S'->S."):
12                     ACTION[i][len(VT)-1] = "acc"
13                     ind = find_gram(it)
14                     if(ind != -1):
15                         for k in range(len(ACTION[i])):
16                             ACTION[i][k] = "r"+str(ind+1)
17
18             else:
19                 next_item = go(item, y[0])
20                 # print("go(%s, %s)-->%s" % (str(item), y[0], str(next_item)))
21                 ind = is_inItems(next_item)
22                 if(ind != -1): #判断是否写入GOTO
23                     if (y[0] in VT):
24                         j = VT2Int[y[0]]
25                         ACTION[i][j] = "s" + str(ind)
26                     if(y[0] in VN):
27                         j = VN2Int[y[0]]
28                         GOTO[i][j] = ind
29
30         i = i + 1

```

• LR0规约算法

- 遍历输入字符串，对于每一个字符，获取当前状态栈的顶部的状态值，通过查询action表获取的当前的操作是移进、规约还是接受
- 如果当前操作是移进 将新的状态放入状态栈当中 当移入的字符放入符号栈中

写下你的评论...

评论0

赞

...

- 如果接收，则结束

```

1  def stipulations():
2      # 根据LR(0)表进行规约
3      global location
4      print('---分析过程---')
5      print("index\t\t", end='')
6      print('%-20s' % 'Status', end='')
7      print('%-50s' % 'Symbol', end='')
8      print('%-30s' % 'Input', end='')
9      print('Action')
10     for i in range(len(dot_grams)):
11         print('---', end='')
12     print()
13
14     symbol_stack.append('#')
15     status_stack.append(0)
16     while not is_end():
17         now_state = status_stack[-1]
18         input_ch = input_str[location]
19         if(input_ch not in Vs):
20             print("错误字符")
21             return -1
22         output()
23         find = ACTION[now_state][VT2Int[input_ch]]
24
25         if find[0] == 's': # 进入action
26             symbol_stack.append(input_ch)
27             status_stack.append(int(find[1]))
28             location += 1
29             print('action[%s][%s]=s%s' % (now_state, input_ch, find[1]))
30
31         elif find[0] == 'r': # 进入goto
32             num = int(find[1])
33             g = grams[num - 1]
34             right_num = count_right_num(g)
35             #print("\n%s"%g)
36             for i in range(right_num):
37                 status_stack.pop()
38                 symbol_stack.pop()
39                 symbol_stack.append(g[0])
40                 now_state = status_stack[-1]
41                 symbol_ch = symbol_stack[-1]
42                 find = GOTO[now_state][VN2Int.get(symbol_ch, -1)]
43                 if find == -1:
44                     print('分析失败')
45                     return -1
46                 status_stack.append(find)
47                 print('%s' % g)
48             else:
49                 return -1
50     print("\n is done")
51     return 0

```

- 全部代码：

```

1  #- coding: utf-8 -*-
2  # @File : exp3.py
3  # @Author: kimoto
4  # @Date : 2018/12/18
5  # @Desc : 编译原理—LR(0)语法分析器
6
7  # 变量申明
8  ACTION = []
9  GOTO = []
10 grams = [] # 用于存放文法字符串 ["S->CA", "S->ccB", "A->CA", "A->a", "B->ccB", "B->b"]
11 #grams = ["S->CA", "S->ccB", "A->CA", "A->a", "B->ccB", "B->b"]
12 #grams = ["S->A", "S->B", "A->aAb", "A->c", "B->aBb", "B->d"]
13 dot_grams = []
14 VN = []
15 VN2Int = {} # 非终结符映射
16 VT2Int = {} # 终结符映射
17 VT = []

```

写下你的评论...

评论0

赞

...

```
23 status_stack = [] # 状态栈
24 symbol_stack = [] # 符号栈
25 now_state = '' # 栈顶状态
26 input_ch = '' # 栈顶字符
27 input_str = '' # 输入串
28 now_step = 0 # 当前步骤
29
30 # print(grams)
31
32
33 #-----预处理-----#
34
35 # 划分终结符和非终结符
36 def get_v():
37
38     vn_num = 0
39     vt_num = 0
40     for s in grams:
41         x,y = s.split("->")
42         # print(x,y)
43         if(x not in VN):
44             VN.append(x)
45             VN2Int.update({x: vn_num})
46             vn_num = vn_num + 1
47         for v in y:
48             if(v.isupper()):
49                 if(v not in VN):
50                     VN.append(v)
51                     VN2Int.update({v: vn_num})
52                     vn_num = vn_num + 1
53             else:
54                 if(v not in VT):
55                     VT.append(v)
56                     VT2Int.update({v: vt_num})
57                     vt_num = vt_num + 1
58
59     for vn in VN:
60         Vs.append(vn)
61     for vt in VT:
62         Vs.append(vt)
63
64     VT.append("#")
65     VT2Int.update({"#": vt_num})
66     print("得到非终结符集合: " + str(VN))
67     print("得到终结符集合: " + str(VT))
68     print("所有的符号集合" + str(Vs))
69
70
71 def dot_gram():
72     # 为所有产生式加点
73     dot_grams.append("S'-.S")
74     dot_grams.append("S'-.S.")
75
76     for gram in grams:
77         ind = gram.find("->")
78         for i in range(len(gram)-ind-1):
79             tmp = gram[ind+2+i] + "." + gram[ind+2+i:]
80             # print(tmp)
81             dot_grams.append(tmp)
82
83
84 # print(str(dot_grams))
85
86
87 #-----构造DNF代码-----#
88
89 def get_VN_gram(v):
90     # 返回非终结符产生的A->.aBb形式
91     res = []
92     for gram in dot_grams:
93         ind = gram.find("->")
94         if(gram[0]==v and gram[ind+2]=="."):
95             res.append(gram)
96     return res
97
98 # print(get_VN_gram("A"))
99
100
```

写下你的评论...

评论0

赞

...

```
106         CLOSURE.append(it)
107         x, y = it.split(".")
108         if(y == ""):
109             continue
110         v = y[0]
111         if(v in VN):
112             res = get_VN_gram(v)
113             for re in res:
114                 if(re not in CLOSURE):
115                     CLOSURE.append(re)
116
117     return CLOSURE
118
119
120 def is_inItems(new_item):
121     #判断item是否已经存在, 存在返回位置, 不存在返回-1
122     if(new_item == None):
123         return -1
124
125     new_set = set(new_item)
126     num=0
127     for item in items:
128         old_set = set(item)
129         if(old_set == new_set):
130             return num
131         num = num + 1
132
133     return -1
134
135 def go(item, v):
136     #生成并返回下一个item
137     tmp = []
138     for it in item:
139         x, y = it.split(".")
140         if(y!=""):
141             if(y[0] == v):
142                 new_it = x + y[0] + "." + y[1:]
143                 tmp.append(new_it)
144
145     if(len(tmp)!=0):
146         new_item = get_CLOSURE(tmp)
147         #print(tmp)
148         #print("go(item, "+v + ") = " + str(new_item))
149         return new_item
150
151
152 def get_items():
153     #构建item的集合
154
155     # 初始化,生成I0
156     item = []
157     init_s = "S'-.S"
158     item.append(init_s)
159     dot_gram()
160
161     for it in item:
162         v = it[it.find(".")-1]
163         if(v in VN):
164             res = get_VN_gram(v)
165             for re in res:
166                 if(re not in item):
167                     item.append(re)
168
169     # print("I0 is : " + str(item))
170     items.append(item)
171
172     num=0
173     for item in items:
174         for v in Vs:
175             # print("item is %s," % str(item) + "v is %s" % v)
176             new_item = go(item, v)
177
178             # 判断状态不为空, 且不存在于原状态中
179             if (new_item != None):
180                 if (is_inItems(new_item) == -1):
181                     # print("添加了%s" % str(new_item))
182                     items.append(new_item)
```

写下你的评论...

评论0

赞

...

```
189
190     action_len = len(VT)
191     goto_len = len(VN)
192     for h in range(len(items)):
193         ACTION.append([])
194         GOTO.append([])
195         for w1 in range(len(VT)+1):
196             ACTION[h].append(" ")
197         for w2 in range(len(VN)):
198             GOTO[h].append(" ")
199
200
201 def lr_is_legal():
202     # 判别lr是否合法
203     has_protocol = 0 #是否存在规约项目
204     has_shift = 0 #是否存在移进项目
205
206     for item in items:
207         for it in item:
208             x, y = it.split(".")
209             if(y == ""):
210                 if(has_protocol != 0 or has_shift != 0):
211                     return False
212                 has_protocol = 1
213             else:
214                 if(y[0] in VT):
215                     has_shift = 1
216     return True
217
218
219 def find_gram(it):
220
221     x, y = it.split(".")
222     mgram = x+y
223     try:
224         ind = grams.index(mgram)
225         return ind
226     except ValueError:
227         return -1
228
229
230 dot_gram()
231 print(dot_grams[1])
232 print(find_gram(dot_grams[1]))
233
234 def get_lr_table():
235     #构建lr分析表
236     init_lr_table()
237     lr_is_legal()
238     i=0
239     j=0
240     for item in items:
241         for it in item:
242             x, y = it.split(".")
243             if(y==""): #判断是否写入ACTION
244                 if (it == "S'->S."):
245                     ACTION[i][len(VT)-1] = "acc"
246                 ind = find_gram(it)
247                 if(ind != -1):
248                     for k in range(len(ACTION[i])):
249                         ACTION[i][k]="r"+str(ind+1)
250
251             else:
252                 next_item = go(item, y[0])
253                 # print("go(%s, %s)-->%s" % (str(item), y[0], str(next_item)))
254                 ind = is_inItems(next_item)
255                 if(ind != -1): #判断是否写入GOTO
256                     if (y[0] in VT):
257                         j = VT2Int[y[0]]
258                         ACTION[i][j] = "s" + str(ind)
259                     if(y[0] in VN):
260                         j = VN2Int[y[0]]
261                         GOTO[i][j] = ind
262
263                 i = i + 1
264
265     print_lr_table()
266
```

写下你的评论...

 评论0 赞


...


```
272         else:
273             return False
274     else:
275         return False
276
277     return True
278
279 # 输出
280 def output():
281     global now_step, status_stack, symbol_stack, input_str, now_state
282     print('%d\t\t' % now_step, end='')
283     now_step += 1
284     print('%-20s' % status_stack, end='')
285     print('%-50s' % symbol_stack, end='')
286     print('%-30s' % input_str[location:len(input_str)], end='')
287
288 # 统计产生式右部的个数
289 def count_right_num(grammar_i):
290     return len(grammar_i) - 3
291
292
293 def stipulations():
294     # 根据LR(0)表进行规约
295
296     global location
297     print('----分析过程----')
298     print("index\t\t", end='')
299     print('%-20s' % 'Status', end='')
300     print('%-50s' % 'Symbol', end='')
301     print('%-30s' % 'Input', end='')
302     print('Action')
303     for i in range(len(dot_grams)):
304         print('---', end='')
305     print()
306
307     symbol_stack.append('#')
308     status_stack.append(0)
309     while not is_end():
310         now_state = status_stack[-1]
311         input_ch = input_str[location]
312         if(input_ch not in Vs):
313             print("错误字符")
314             return -1
315         output()
316         find = ACTION[now_state][VT2Int[input_ch]]
317
318         if find[0] == 's': # 进入action
319             symbol_stack.append(input_ch)
320             status_stack.append(int(find[1]))
321             location += 1
322             print('action[%s][%s]=s%s' % (now_state, input_ch, find[1]))
323
324         elif find[0] == 'r': # 进入goto
325             num = int(find[1])
326             g = grams[num - 1]
327             right_num = count_right_num(g)
328             #print("\n%s"%g)
329             for i in range(right_num):
330                 status_stack.pop()
331                 symbol_stack.pop()
332             symbol_stack.append(g[0])
333             now_state = status_stack[-1]
334             symbol_ch = symbol_stack[-1]
335             find = GOTO[now_state][VN2Int.get(symbol_ch, -1)]
336             if find == -1:
337                 print('分析失败')
338                 return -1
339             status_stack.append(find)
340             print('%s' % g)
341         else:
342             return -1
343
344     print("\n is done")
345     return 0
346
347
348
349
```

```
355     num = 1
356     for gram in grams:
357         print("(%d)%s"%(num, str(gram)))
358         num = num + 1
359
360 def print_items():
361
362     print("----状态集合----")
363     num=0
364     for it in items:
365         print("(%d)%s"%(num, str(it)))
366         num = num + 1
367
368 def print_lr_table():
369     # 表头
370     print('----LR分析表----')
371     print('\t\t\t\t', end='')
372     print((' %4s' % '') * (len(VT) - 3), end='')
373     print('Action', end='')
374     print((' %4s' % '') * (len(VT) - 3), end='')
375     print('\t\t\t\t', end='')
376     print((' %3s' % '') * (len(VN) - 2), end='')
377     print('GOTO', end='')
378     print((' %3s' % '') * (len(VN) - 2), end='')
379     print('\t\t\t\t')
380     print('\t\t\t\t', end='')
381     for i in VT:
382         print(' %3s\t' % i, end='')
383     print('\t\t\t\t', end='')
384     k = 0
385     for i in VN:
386         print(' %3s\t' % i, end='')
387     print('\t\t\t\t')
388     for i in range(len(dot_grams)):
389         print('---', end='')
390     print()
391     # 表体
392     for i in range(len(items)):
393         print('%5d\t\t\t' % i, end='')
394         for j in range(len(VT)):
395             print(' %4s' % ACTION[i][j], end='')
396             print('\t\t\t\t', end='')
397         for j in range(len(VN)):
398             if not GOTO[i][j] == -1:
399                 print(' %4s' % GOTO[i][j], end='')
400             else:
401                 print('\t\t\t\t', end='')
402             print('\t\t\t\t')
403     for i in range(len(dot_grams)):
404         print('---', end='')
405     print()
406
407
408 if __name__ == '__main__':
409
410
411     if(len(grams)==0):
412         with open("1.txt", "r") as f:
413             for line in f:
414                 line = line.replace('\n', "")
415                 grams.append(line)
416             f.close()
417
418
419     get_v()      # 分割终结符和非终结符
420     print_grams() # 输出文法产生式
421     get_items()  # 生成状态集合
422     print_items() # 输出状态集合
423     get_lr_table() # 生成lr分析表
424     input_str = "abaaabaaab#" # 待分析字符串
425     stat = stipulations() # 规约
426     if(stat == 0):
427         print("\n %s 符合文法规则" % input_str)
428     else:
429         print("\n %s 不符合文法规则" % input_str)
```


 0人点赞 >



 编译原理



"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下



kimotoo

总资产1 (约0.11元) 共写了2065字 获得1个赞 共1个粉丝

关注

- mes生产系统
- 补一颗牙齿要多少钱
- mes 模块
- 生产mes系统
- python
- 翻译英语中文
- 夏恩英语怎么
- 老化 试验箱
- kvm技术
- 英语手写信
- 英语棕色怎么读
- mes制造系统
- 开发mes系统
- mes 智能制造
- 英语协议翻译
- pda RFID读写
- ccnp培训班
- mes生产系统

写下你的评论...

 评论0  赞 ...