นาย กวิน เหลี่ยววงค์ภูธร 5731004721

Project “K2W” Language 1.0.0

ภาษา K2W ที่ได้ทำขึ้นนี้มีต้นแบบมาจาก ภาษา Python แต่ได้มีการปรับเปลี่ยน และ ลดทอนความสามารถบางประการออกไป ทั้งนี้เพื่อให้เหมาะสมกับระยะเวลา และการใช้คำพูดในการบรรยายมากขึ้น

ในขณะนี้ ภาษา K2W มีความสามารถดังนี้

1. การ define function ทั้งมี และไม่มี parameter
2. การเรียกใช้ function ทั้งแบบ ส่งและไม่ส่ง argument โดย argument เป็นได้เพียง ตัวแปรเท่านั้น
3. การ assign ค่า ให้ตัวแปร โดยค่าที่สามารถใช้ได้นั้น เป็นทั้งค่าคงที่ สมการ หรือค่าจากการเรียกใช้ function
4. การเรียกใช้ คำสั่ง return
5. การเรียกใช้ฟังก์ชัน หรือประกาศตัวแปรใช้ได้เพียงคำ ที่เป็น keyword เท่านั้น คือคำว่า variable, function และ func_name ทั้งนี้จะพัฒนาในโอกาสต่อไป

โดย Gramma ของ ภาษา K2W นั้น มีดังนี้

1. CODE → define OBJ MORE_CODE
2. MORE_CODE → CODE | λ
3. OBJ → function PRAM start FUNC end
4. FUNC → INLINE MORE_LINE
5. MORE_LINE → FUNC | λ
6. INLINE → ASSIGNMENT | USE_FUNC newline |return VALUE newline
7. ASSIGNMENT → assign VALUE to variable newline
8. USE_FUNC → use func_name ARG
9. ARG → with variable MORE_VAR | λ
10. PRAM → receive variable MORE_VAR as parameter | λ
11. MORE_VAR → variable MORE_VAR | λ
12. VALUE → TERM EQ | USE_FUNC EQ
13. EQ → OP VALUE | λ
14. TERM → variable | number
15. OP → add | subtract | multiply | divide

ตัวอย่าง ภาษา

```
1  define function receive variable variable variable variable as parameter start
2      assign use func_name with variable variable to variable newline
3      return variable add variable newline
4  end
5  define function start
6      use func_name newline
7      assign variable multiply use func_name with variable variable variable add variable to variable newline
8  end
9
```

จาก Gramma ข้างต้น สามารถเขียน first set และ follow set ได้ดังนี้

nonterminal	**First set**	**Follow set**
CODE	define	$
MORE_CODE	define, λ	$
OBJ	function	define, $
FUNC	assign, use, return	end
MORE_LINE	assign, use, return, λ	end
INLINE	assign, use, return	return, assign, use, end
ASSIGNMENT	assign	return, assign, use, end
USE_FUNC	use	add, subtract, multiply, divide, newline, to
ARG	with, λ	add, subtract, multiply, divide, newline, to
PRAM	receive, λ	start
MORE_VAR	variable, λ	as, add, subtract, multiply, divide, newline, to
VALUE	variable, number, use	to, newline
EQ	add, subtract, multiply, divide, λ	to, newline
TERM	variable, number	add, subtract, multiply, divide, newline, to
OP	add, subtract, multiply, divide	variable, number, use

เราสามารถ แยก Gramma ออกมา เป็น กฎ ข้อย่อย ๆ เพื่อที่จะนำไปสร้าง Parsing Tables ได้จำนวน 28 ข้อดังนี้

1. CODE → define OBJ MORE_CODE
2. MORE_CODE → CODE
3. MORE_CODE → λ
4. OBJ → function PRAM start FUNC end
5. FUNC → INLINE MORE_LINE
6. MORE_LINE → FUNC
7. MORE_LINE → λ
8. INLINE → ASSIGNMENT
9. INLINE → USE_FUNC newline
10. INLINE → return VALUE newline
11. ASSIGNMENT → assign VALUE to variable newline
12. USE_FUNC → use func_name ARG
13. ARG → with variable MORE_VAR
14. ARG → λ
15. PRAM → receive variable MORE_VAR as parameter
16. PRAM → λ
17. MORE_VAR → variable MORE_VAR
18. MORE_VAR → λ
19. VALUE → TERM EQ
20. VALUE → USE_FUNC EQ
21. EQ → OP VALUE
22. EQ → λ
23. TERM → variable
24. TERM → number
25. OP → add
26. OP → subtract
27. OP → multiply
28. OP → divide

เมื่อ เรานำไปเขียนกฎ เป็น ตาราง จะมีรูปร่างดังตารางต่อไปนี้

	define	function	start	end	newline	Return	assign	to	variable	use	func_name
CODE	1										
MORE_CODE	2										
OBJ		4									
FUNC						5	5			5	
MORE_LINE				7		6	6			6	
INLINE						10	8			9	
ASSIGNMENT							11				
USE_FUNC										12	
ARG					14			14			
PRAM			16								
MORE_VAR					18			18	17		
VALUE									19	20	
EQ					22			22			
TERM									23		
OP											

	with	receive	as	parameter	number	add	subtract	multiply	divide	$
CODE										
MORE_CODE										3
OBJ										
FUNC										
MORE_LINE										
INLINE										
ASSIGNMENT										
USE_FUNC										
ARG	13					14	14	14	14	
PRAM		15								
MORE_VAR			18			18	18	18	18	
VALUE					19					
EQ						21	21	21	21	
TERM					24					
OP						25	26	27	28	

ใน project นี้ได้มีการ เขียน parser สำหรับภาษา K2W ด้วยภาษา Python โดยมี core code ดังต่อไปนี้

```
54  ##################### check Gramma ##########################
55
56  r = 1
57  token_index = 0
58  accept = False
59  check_stack = ['$','CODE']
60  while True:
61      print('word ',r)
62      top_stack = check_stack.pop()
63      now_token = token_data[token_index]
64      print(top_stack+' '+now_token)
65      if(top_stack=='LAMBDA'):
66          continue
67      if(token_index >= len(token_data) or top_stack =='error'):
68          break
69      if(top_stack =='$'):
70          if(now_token=='$'):
71              accept = True
72              break
73          else:
74              break
75      elif top_stack  == now_token:
76          r+=1
77          token_index+=1
78          print('==')
79      else:
80          try:
81              rule_number = table[top_stack][now_token]
82              print('rule nember',rule_number)
83              l = len(rule[rule_number])
84              for i in range(l):
85                  check_stack.append(rule[rule_number][l-1-i])
86          except KeyError:
87              break
88      print(check_stack)
89  ###################################################################
90  ################### print answer ################################
91  if(accept):
92      print('ACCEPT')
93  else:
94      print('ERROR')
95  ###################################################################
```

สำหรับ Code ฉบับเต็ม และการพัฒนาสามารถติดตามเพิ่มเติมได้ที่ https://github.com/KawinL/K2W

ผลลัพธ์เมื่อ input ถูก Gramma

TERMINAL

```
('word ', 43)
to to
==
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable']
('word ', 44)
variable variable
==
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline']
('word ', 45)
newline newline
==
['$', 'MORE_CODE', 'end', 'MORE_LINE']
('word ', 46)
MORE_LINE end
('rule nember', 7)
['$', 'MORE_CODE', 'end', 'LAMBDA']
('word ', 46)
LAMBDA end
('word ', 46)
end end
==
['$', 'MORE_CODE']
('word ', 47)
MORE_CODE $
('rule nember', 3)
['$', 'LAMBDA']
('word ', 47)
LAMBDA $
('word ', 47)
$ $
ACCEPT
```

ผลลัพธ์เมื่อ input ผิด Gramma

TERMINAL

```
('word ', 16)
MORE_VAR variable
('rule nember', 17)
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable', 'to', 'EQ', 'MORE_VAR', 'variable']
('word ', 16)
variable variable
==
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable', 'to', 'EQ', 'MORE_VAR']
('word ', 17)
MORE_VAR to
('rule nember', 18)
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable', 'to', 'EQ', 'LAMBDA']
('word ', 17)
LAMBDA to
('word ', 17)
EQ to
('rule nember', 22)
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable', 'to', 'LAMBDA']
('word ', 17)
LAMBDA to
('word ', 17)
to to
==
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline', 'variable']
('word ', 18)
variable variable
==
['$', 'MORE_CODE', 'end', 'MORE_LINE', 'newline']
('word ', 19)
newline newlinej
ERROR
```

แนวทางการพัฒนาต่อ

- เนื่องจากขณะนี้ ภาษา K2W ไม่ตรงตามหลัก Gramma ภาษาอังกฤษ จึงทำให้ไม่สามารถ ใช้ library speech recognition ที่มีอยู่แล้วได้เนื่องจาก การทำ speech recognition นั้นจะใช้ การทำ data analytic บนข้อมูล ประโยคตัวอย่างภาษานั้น ๆ ร่วมกันกับการฟังเสียง มีแนวทางแก้ไขอยู่ 2 วิธี
    - ปรับปรุง Gramma ให้ เข้ากับภาษา อังกฤษมากขึ้น
    - จัดสร้างเครื่องมือ speech recognition เอง เพื่อให้ได้ตาม Gramma ตามภาษาของเรา ขณะนี้ได้ลอง อ่านตัว open source ชื่อ CMUSphinx
- ปรับปรุงให้สามารถ รับค่าชื่อ ต่าง ๆ ได้
- จัดสร้างตัวแปลง ไปเป็นภาษา Python โดยในตอนนี้แนวคิดหลักคือ คำว่า start เป็นคำที่แทน “:” พร้อมกับการขึ้นบรรทัดใหม่และเพิ่มย่อหน้า ส่วน end เป็นการ ลดย่อหน้าลง
- เพิ่มคำสั่ง control flow เช่น if else for เป็นต้น