Lexer:

a. MSGNAME_REGEX: [A-Z][a-zA-Z0-9]*
   WS_REGEX: "ws://[0-9]*[a-zA-Z][a-zA-Z0-9]*(.[a-zA-Z0-9]+)+/([A-F0-9]+|[a-f0-9]+)"
   STRING_REGEX: "[a-zA-Z0-9]*"
   NUMBER_REGEX: (-[1-9][0-9]*|[1-9][0-9]*|0)
   FIELDNAME_REGEX: w|w[a-rt-zA-Z][a-zA-Z]*|ws[a-zA-Z]+|[a-vxyzA-Z][a-zA-Z]*

b.

```
begin match:
Message Name: Msg
Begin body {
Custom Field: question
Value (Str): "Are labs in compilers full of fun"
End custom field ;
WS Field: WS : "ws://compilers.ipads.sjtu.edu.cn/DEADBEEF"
End ws field;
End msg

begin match:
Message Name: Msg1
Begin body {
Custom Field: Error
Value (Str): "Wins come all day under Boss ZHAN BIANYI"
End custom field ;
WS Field: Error: Invalid ws string format

begin match:
Message Name: QuizRes
Begin body {
Custom Field: answer
Value (Str): "Your Answer is Always Correct"
End custom field ;
Custom Field: score
Value (Num): 100
End custom field ;
WS Field:  WS : "ws://compilers.ipads.sjtu.edu.cn/1234abcd"
End ws field;
End msg
```

c. M = “msg” W = “w”
   Click Here!

`

Grammar:

1. 图中 E’ 替换为 S 即可

Click Here!

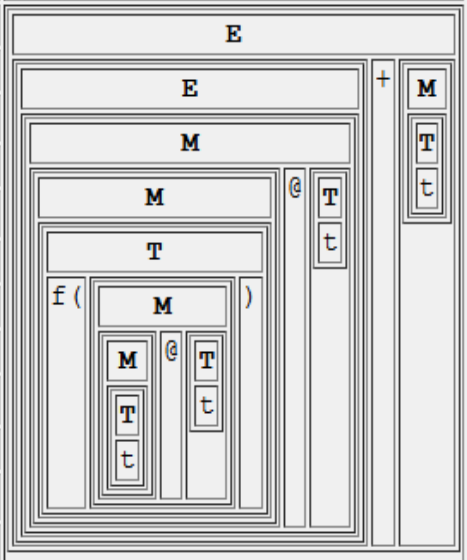
2.

LR table										
State	ACTION						GOTO			
	+	@	f(	)	t	$	S	E	M	T
0			s4		s5			1	2	3
1	s6					acc				
2	r_2	s7				r_2				
3	r_4	r_4				r_4				
4			s10		s11				8	9
5	r_6	r_6				r_6				
6			s4		s5				12	3
7			s4		s5					13
8		s15		s14						
9		r_4		r_4						
10			s10		s11				16	9
11		r_6		r_6						
12	r_1	s7				r_1				
13	r_3	r_3				r_3				
14	r_5	r_5				r_5				
15			s10		s11					17
16		s15		s18						
17		r_3		r_3						
18		r_5		r_5						

3.

Trace			
Step	Stack	Input	Action
1	0	f(t @ t) @ t + t $	s4
2	0 f(4	t @ t) @ t + t $	s11
3	0 f(4 t 11	@ t) @ t + t $	r_6
4	0 f(4 T	@ t) @ t + t $	9
5	0 f(4 T 9	@ t) @ t + t $	r_4
6	0 f(4 M	@ t) @ t + t $	8
7	0 f(4 M 8	@ t) @ t + t $	s15
8	0 f(4 M 8 @ 15	t) @ t + t $	s11
9	0 f(4 M 8 @ 15 t 11	) @ t + t $	r_6
10	0 f(4 M 8 @ 15 T	) @ t + t $	17
11	0 f(4 M 8 @ 15 T 17	) @ t + t $	r_3
12	0 f(4 M	) @ t + t $	8
13	0 f(4 M 8	) @ t + t $	s14
14	0 f(4 M 8) 14	@ t + t $	r_5
15	0 T	@ t + t $	3
16	0 T 3	@ t + t $	r_4
17	0 M	@ t + t $	2
18	0 M 2	@ t + t $	s7
19	0 M 2 @ 7	t + t $	s5
20	0 M 2 @ 7 t 5	+ t $	r_6
21	0 M 2 @ 7 T	+ t $	13
22	0 M 2 @ 7 T 13	+ t $	r_3
23	0 M	+ t $	2
24	0 M 2	+ t $	r_2
25	0 E	+ t $	1
26	0 E 1	+ t $	s6
27	0 E 1 + 6	t $	s5
28	0 E 1 + 6 t 5	$	r_6
29	0 E 1 + 6 T	$	3
30	0 E 1 + 6 T 3	$	r_4
31	0 E 1 + 6 M	$	12
32	0 E 1 + 6 M 12	$	r_1
33	0 E	$	1
34	0 E 1	$	acc

Tree



Semantic Analysis:

1. E0: {target -> int}
   E1: E0 + {totalSum -> int, res -> int}
   E2: E1 + {limit -> int}
   E3: E2 + {p1 -> Point}
   E4: E3 + {val -> int}
   E5: E4 - {limit -> int} + {limit -> int}
   E6: E3
   E7: E3 + {x -> int}
   E8: E3
   E9: E3 + {i -> int}
   E10: E3
   E11: E1
   E12: E11 + {msg -> string}
   E13: E1
2. 4 : E1
   16 : E5
   27 : E9
   42 : E12
   46 : E13