CPSC 323 Project 2 Documentation

In project 2, we had to create a syntax analyzer program, in which we give it some strings to parse and check to see if it is accepted and the output shows the stack before action, the state it is at, the symbol in the state, the action its going to perform, and if there is a reduce action, it prints the stack before reduction, and stack after reduction. The program concatenataes the “$” to the end of the input string before it starts to parse the input and it appends the “$” symbol and 0 to the stack, while the parser is parsing through the string inputs, it prints the stack whenever it does any actions. There is a constructor in the python program in which has a stack initialized and has the parse table initialized, there is a loop which runs up until all the the elements in the input string have been parsed, it checks to see if the state and symbol are in the parse table and if it isn’t it then says the string is not accepted, it checks the last element of the list for the state and second to last element for the symbol and if they aren’t in the parse table it returns string is not accepted, if the symbol and state are in the symbol table, it checks to see what action or goto it is if it a shift action, then it appends the symbol and state to the end of the list, if the action is s7, then it appends 7 at the end of the list. If it is a reduce action, then it checks for the last index of the action, and converts it to an int to check to see if there is a reduce rule for it, if the action is r4, it calls the function reduce_rule_4 which replaces one of the values in the stack with the functions and the written code inside it function, since reduce_rule_4 reudces F from T(T→F), it replaces F with T and the number ahead of it, based on the value of the goto. There are so many reduce functions for specific reduction rules in which it replaces the out value with the in value in the stack, at the end of the program, the parser object is created and there is a list of values containing the three input strings that we tested in the program, the three input strings are "(id+id)*id", "id*id", and "(id*)", we loop through the list and have it print the input and the output, the output is the process of parsing through the input string which prints out the stack and all the actions that were happening during the parsing of the input.

Here is what the program outputs

Test example 1

Input: (id+id)*id
Stack before action: ['$', 0]
State: 0
Symbol: (
Action: s4
Stack before action: ['$', 0, '(', 4]

State: 4
Symbol: id
Action: s5
Stack before action: ['$', 0, '(', 4, 'id', 5]
State: 5
Symbol: +
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1]
State: 1
Symbol: +
Action: s6
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6]
State: 6
Symbol: id
Action: s5
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'id', 5]
State: 5
Symbol:)
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'F', 1, '+', 6, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8]
State: 8
Symbol:)
Action: s11
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11]
State: 11
Symbol: *
Action: r5
Action while reduction: r5
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2]
State: 2
Symbol: *
Action: s7
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7]
State: 7
Symbol: id
Action: s5

Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'id', 5]
State: 5
Symbol: $
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1]
State: 1
Symbol: $
Action: acc
Output: String is accepted.

Test example 2

Input: id*id
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0]
State: 0
Symbol: id
Action: s5
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'id', 5]
State: 5
Symbol: *
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2]
State: 2
Symbol: *
Action: s7
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7]
State: 7
Symbol: id
Action: s5
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'id', 5]

State: 5
Symbol: $
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1]
State: 1
Symbol: $
Action: acc
Output: String is accepted.

Test example 3

Input: (id*)
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0]
State: 0
Symbol: (
Action: s4
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0, '(', 4]
State: 4
Symbol: id
Action: s5
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0, '(', 4, 'id', 5]
State: 5
Symbol: *
Action: r6
Action while reduction: r6
Stack before reduction: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0, '(', 4, 'id', 5]
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0, '(', 4, 'F', 2]
State: 2

Symbol: *
Action: s7
Stack before action: ['$', 0, '(', 4, 'F', 1, '+', 6, 'F', 8, ')', 11, 'F', 2, '*', 7, 'F', 1, '$', 0, 'F', 2, '*', 7, 'F', 1, '$', 0, '(', 4, 'F', 2, '*', 7]
Output: String is not accepted.