**Forward chaining Algorithm**

**Define the Following Functions:**

1. **search\_cvl** (*double* *variable*): this function will search for an entry in the clause variable list and find the entry that matches the argument variable and return the clause number, Ci, corresponding to the matching entry. Then first call **update\_VL** (*Ci*).Thencall **clause\_to\_rule** (*Ci*):
2. **clause\_to\_rule** (*integer* *variable*): - This function will convert Clause number, Ci, to rule number, Ri, using the following formula. After computing the value of Ri, it will call **validate\_Ri** (*Ri*):

If the clause numbers in the clause variable list are sequenced like 1,2,3,4,5, ……), the formula is:

Rule # = [({Quotient (Clause # / 4)} +1)]

If the rule numbers are sequenced like 10,20,30,40,50, ……), the formula is:

Rule # = [({Quotient (Clause # / 4)} +1) \* 10]

It has been assumed that four slots have been assigned for each rule in the Clause Variable list. If other than four has been assigned replace 4 by that number.

1. **update\_VL** (*integer* *variable*)**:** For each variable (maximum of four) starting location Ci**,** It will check if the variable is instantiated in the variable list or not. If not, it will ask the user to provide the values of variables and instantiate them.
2. **validate\_Ri** (*integer* *variable*): It will check if the values of the variable in the ‘if’ clauses of the rule, Ri, are satisfied with the values in the variable list. If they do, add the conclusion of the rule to the global derived conclusions list as well as to the Global Conclusion Variable Queue and return.
3. **process** **(variable)** 
   1. Instantiate the value of the variable in the variable list. Call **search\_cvl**(*variable*)
   2. return
4. **FC\_main function**

**Declaration of the FC main function**

* Write functions’ prototypes
* Declare variables and arrays.
* Create a list of rules. It may need some organization. You have to determine which format you want to use for efficient processing by the **Validate\_Ri** function.
* Create a Variable List as an array.
* Create a Clause Variable list as an array.
* Create a Global Conclusion variable queue as an array.

**Processing of the main function:**

* 1. Read the value of the given variable (the variable whose value is provided) and **call process** **(variable)**
  2. If the Global Conclusion Variable Queue is not empty, in a loop call **process (variable) for** each variable in the Global Conclusion Variable Queue which occurs in the clause variable list. After the function returns, it will delete the variable from the Global Conclusion Variable Queue.
  3. Print all the derived conclusions from the Derived Conclusion List

Example of the Global Derived Conclusion list

|  |
| --- |
| Interest = FALL |
| Stock = Rise |
|  |