Just some concepts regarding rule framing in the smt-lib format -

1: If (((100 < m < 200) or (100 < n < 200)) and (((300<m<400) or(300<n<400)))

(define-fun rule1_applies () Bool (and (or (range m 100 200) (range n 100 200))
(or (range m 300 400) (range n 300 400))))

In the above case, we have 2 clauses and each clause has 2 predicates -

Clause 1 : **((100 < m < 200) or (100 < n < 200))**
Clause 2 : **((300<m<400) or (300<n<400))**

Concept 1: Now, all the predicates in a given specific clause shall be connected by an or -

For eg - Clause 1 has two predicates and both are connected by a 'or' clause.

Concept 2: Now all the clauses must be connected by a 'and' clause

For eg - **(((100 < m < 200) or (100 < n < 200))** **and** **(((300<m<400) or(300<n<400)))**

2 : Let us consider a rule whose conditions are of the form -

If (((100 < m < 200) or (100 < n < 200)) and (((300<m<400) or(300<n<400)))
and(((200<m<400) or(350<n<400))

Its equivalent smt format shall be -

(define-fun rule1_applies () Bool (and (or (range m 100 200) (range n 100 200))
(or (range m 300 400) (range n 300 400))
(or (range m 200 400) (range n 170 600))))

Now, the logic is -

For those connected by and or statement, all the variables shall be in the same table and then the others must be in the other tables and all of these must be preceded by an 'or' statement. And the beginning must have an 'and' statement to connect all of the statements.

3. Consider another rule -

1: If 100<m<120 then output = 200

(define-fun rule1_applies () Bool (and (range m 100 120)))

The and must be in the rule at the beginning even if it is a single predicate.