1 Main idea

We initialize all literals to $False$, which naturally satisfy all "$False$-*must-exist*" conditions. However, at this time, there may exist some "*lead-to*" conditions not satisfied, and we only change the value of a literal when necessary. Thus it means that we turn some literals from $False$ to $True$ only when we don't have any other choice, and we cannot turn some literals from $True$ to $False$ since it will make some *lead-to* condition unsatisfied.
When a *lead-to* condition is detected to be unsatisfied, it must have the form of $True \Rightarrow False$. And since we cannot turn some literals from $True$ to $False$, the only action we can take is to turn the literal at the right side of the arrow from $False$ to $True$.
We iteratively check all *lead-to* conditions, and only change literals when necessary. This process must eventually terminate, because we can simply set all literals to $True$. When it terminates, all *lead-to* conditions are satisfied and all literals which are reset to $True$ cannot be changed again. Then we check at this time whether all $False$-*must-exist* conditions are satisfied, and if so, we successfully find a satisfying solution. Otherwise, there is no satisfying solution since that we cannot change any literals to get all conditions satisfied (change some literals from $True$ to $False$ will make some *lead-to* condition unsatisfied, and change some literals from $False$ to $True$ will not make more $False$-*must-exist* conditions satisfied).

2 Pseudo code

Algorithm 1 Find satisfying solution

```
Initialize all literals x_1, x_2, ..., x_n to False
/* Initialize all literals x_j of form ⇒ x_j to True */
for T_i in L = {T_1, T_2, ..., T_P} do
  if T_i has the form: ⇒ x_j then
    set x_j to True
  end if
end for
/* Iteratively satisfy all lead-to conditions */
repeat
  for T_i in L = {T_1, T_2, ..., T_P} do
    if T_i is satisfied then
      continue
    else
      set the literal at the right side of "⇒" to True
    end if
  end for
until no literal is changed
/* Check whether all False-must-exist conditions are satisfied */
for F_i in F = {M_1, M_2, ..., M_Q} do
  if F_i is not satisfied then
    return no satisfying solution exists
  end if
end for
return x_1, x_2, ..., x_n as a satisfying solution
```

3 Proof of correctness

We firstly prove by induction that we only change a literal from $False$ to $True$ when necessary, which means that if we do not turn it from $False$ to $True$ then some *lead-to* conditions cannot be satisfied.
To prove: For any k, if a *lead-to* condition $(x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_k}) \Rightarrow x_j$ is not satisfied, we can only change x_j from $False$ to $True$.
Base case: When $k = 0$, we know that the *lead-to* condition has the form of $\Rightarrow x_j$, then we have no choice but to change x_j to $True$. Thus the case of $k = 0$ holds.
Induction hypothesis: Assume that when $k = n$, to satisfy the *lead-to* condition $(x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_k}) \Rightarrow x_j$ which is not satisfied now, we have no choice but to change x_j to $True$.
Induction step: When $k = n + 1$, if we detect a *lead-to* condition $(x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_{k+1}}) \Rightarrow x_j$ is unsatisfied, then we have $x_{i_1} = x_{i_2} = ...x_{i_{k+1}} = True$ and $x_j = False$. By the hypothesis we know that we cannot change any of

x_{i_1},x_{i_2},...$x_{i_{k+1}}$, so the only action we can take is to change x_j from $False$ to $True$. Thus, the proposition holds for $k = n + 1$.
In conclusion, the proposition holds, which means that we only change literals from $False$ to $True$ when necessary and attempting to change literals from $True$ to $False$ will make some conditions unsatisfied.

After iteratively checking all *lead-to* conditions to make them satisfied, we start to check all $False$-*must-exist* conditions. If all of them are satisfied, then we have find a solution that satisfies all the $P + Q$ conditions. Otherwise, there exists a $False$-*must-exist* condition is not satisfied, say $\overline{x_{i_1}} \vee \overline{x_{i_2}} \vee ... \vee \overline{x_{i_k}}$. Then we know that all literals in this condition are $True$, that is, $x_{i_1} = x_{i_2} = ...x_{i_k} = True$. However, as proved above, we cannot change any literals that are set to $True$, thus this $False$-*must-exist* condition can never be satisfied, which means that there is no satisfying solution.

In summary, the literals we set to $True$ can only be set to $True$, otherwise some *lead-to* conditions will fail. After doing that, if any $False$-*must-exist* condition fails, then we can never satisfy it, thus there is no solution. Otherwise, if each $False$-*must-exist* is satisfied, we find a solution.

4 Time complexity

Let n denote the number of literals, P denote the number of *lead-to* conditions, Q denote the number of $False$-*must-exist* conditions, k denote the maximum number of literals to the left of "$\Rightarrow$", m denote the maximum number of literals involved in $False$-*must-exist* conditions.
To initialize all literals to $False$, it takes $O(n)$.
To set all literals x_j of form $\Rightarrow x_j$ to $True$, it takes $O(P)$.
The iterative process will perform at most $O(\min(n, P))$ times since each iteration we at least change one literal and satisfy one more condition. At each iteration, we will check all *lead-to* conditions, and there are $O(P)$ such conditions. To check each *lead-to* condition, we need check all literals in it, which costs $O(k)$. Thus the cost of iterative process is $O(\min(n, P)kP)$.
To check whether all $False$-*must-exist* conditions are satisfied, we need check $O(Q)$ conditions. And to check each condition, we need visit all literals involved in the condition, which costs $O(m)$. Thus it takes $O(mQ)$ to check all $False$-*must-exist* conditions.

In summary, the total complexity is $O(n+\min(n, P)kP + mQ)$.