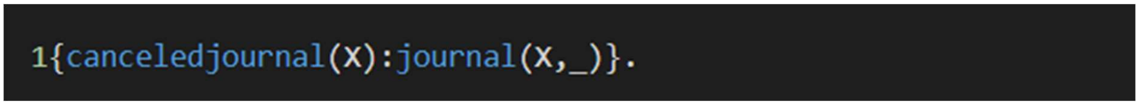
# CMT304 – Part 1: Logic Programming

Mohammed Shaad Mehboob Matcheswala(21065793)

CARDIFF UNIVERSITY

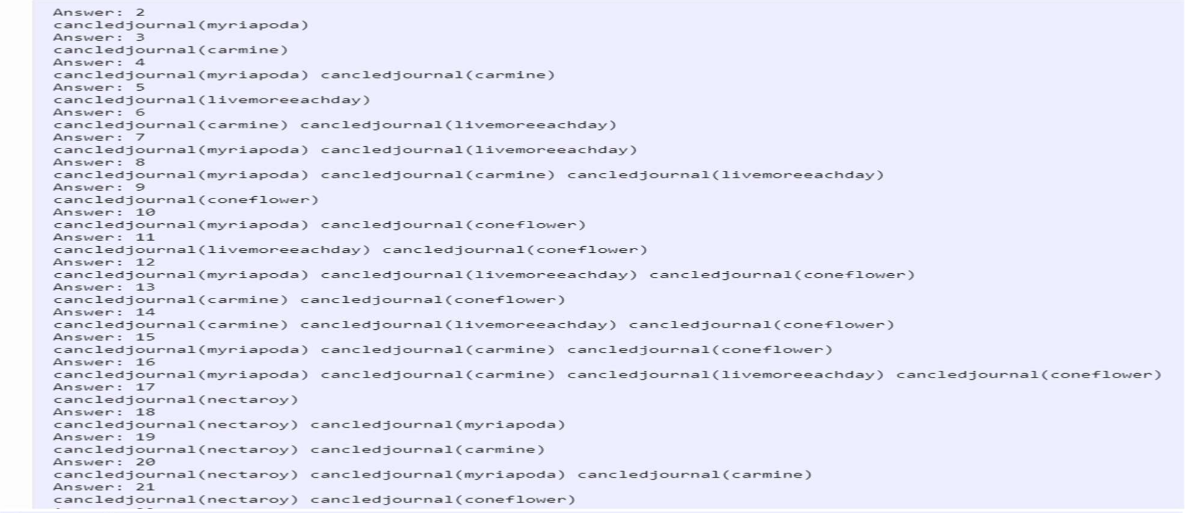
1]

In guess and test methodology the guess step will generate all the possible answer sets. In this particular example, the guess rule is applied to the journals and is sent to the list of canceledjournal. Here, all the possible sets are generated for the canceledjournal, and for 5 journals it generated 31 sets.

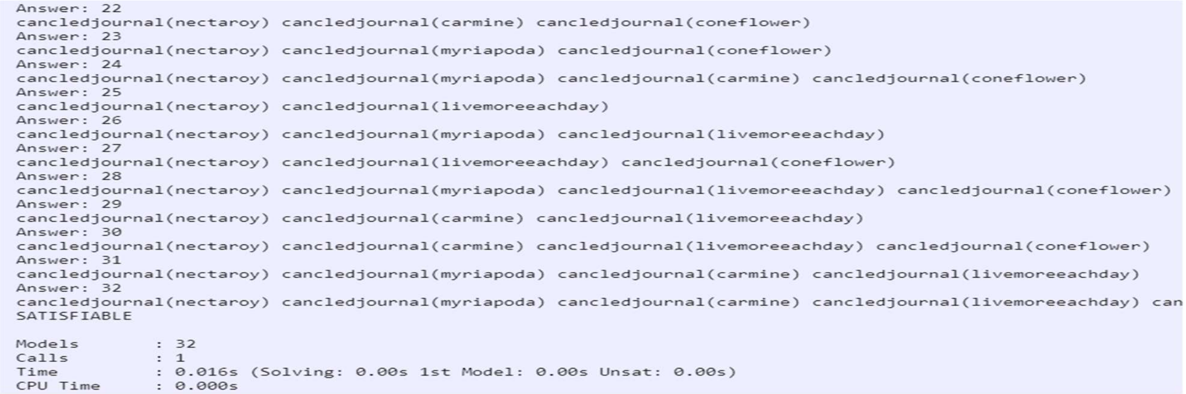


```
1{canceledjournal(X):journal(X,_)}.
```

For example, the list of all the sets generated is displayed in the below figure.



```
Answer: 2
cancledjournal(myriapoda)
Answer: 3
cancledjournal(carmine)
Answer: 4
cancledjournal(myriapoda) cancledjournal(carmine)
Answer: 5
cancledjournal(livemoreeachday)
Answer: 6
cancledjournal(carmine) cancledjournal(livemoreeachday)
Answer: 7
cancledjournal(myriapoda) cancledjournal(livemoreeachday)
Answer: 8
cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(livemoreeachday)
Answer: 9
cancledjournal(coneflower)
Answer: 10
cancledjournal(myriapoda) cancledjournal(coneflower)
Answer: 11
cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 12
cancledjournal(myriapoda) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 13
cancledjournal(carmine) cancledjournal(coneflower)
Answer: 14
cancledjournal(carmine) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 15
cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(coneflower)
Answer: 16
cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 17
cancledjournal(nectaroy)
Answer: 18
cancledjournal(nectaroy) cancledjournal(myriapoda)
Answer: 19
cancledjournal(nectaroy) cancledjournal(carmine)
Answer: 20
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(carmine)
Answer: 21
cancledjournal(nectaroy) cancledjournal(coneflower)
```

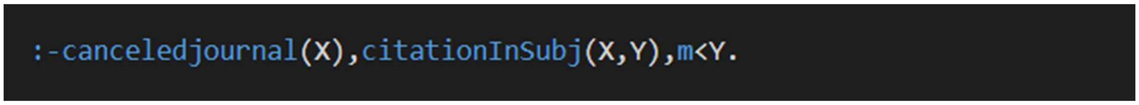


```
Answer: 22
cancledjournal(nectaroy) cancledjournal(carmine) cancledjournal(coneflower)
Answer: 23
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(coneflower)
Answer: 24
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(coneflower)
Answer: 25
cancledjournal(nectaroy) cancledjournal(livemoreeachday)
Answer: 26
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(livemoreeachday)
Answer: 27
cancledjournal(nectaroy) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 28
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 29
cancledjournal(nectaroy) cancledjournal(carmine) cancledjournal(livemoreeachday)
Answer: 30
cancledjournal(nectaroy) cancledjournal(carmine) cancledjournal(livemoreeachday) cancledjournal(coneflower)
Answer: 31
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(livemoreeachday)
Answer: 32
cancledjournal(nectaroy) cancledjournal(myriapoda) cancledjournal(carmine) cancledjournal(livemoreeachday) can
SATISFIABLE

Models       : 32
Calls        : 1
Time         : 0.016s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

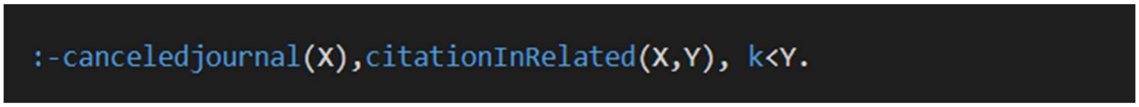
In the test rule, we apply constraints to eliminate all the unwanted values from the answer set.

In the first test rule, all the answer sets from the canceledjournal are eliminated whose citation count in the subject area of a particular journal (Y) is greater than m (citation count in subject).



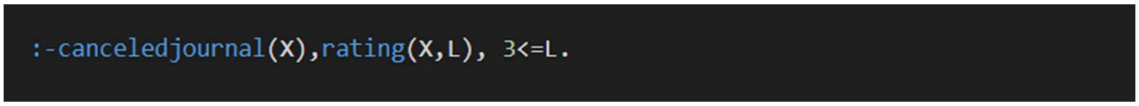
```
:-canceledjournal(X),citationInSubj(X,Y),m<Y.
```

In the second test rule, all the remaining answer sets from the canceledjournal are taken into consideration and elimination is performed on the sets whose citation count in the related area per journal (Y) is greater than k (citation count in a related area).



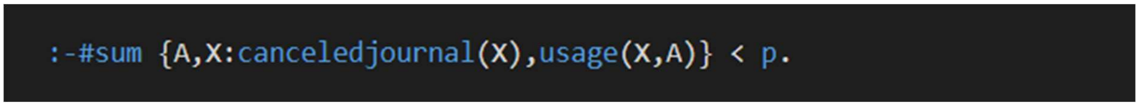
```
:-canceledjournal(X),citationInRelated(X,Y), k<Y.
```

Similarly, in the third test rule, the aim is to apply the constraints that will delete all the sets from the canceledjournal whose ratings are greater than or equal to 3.



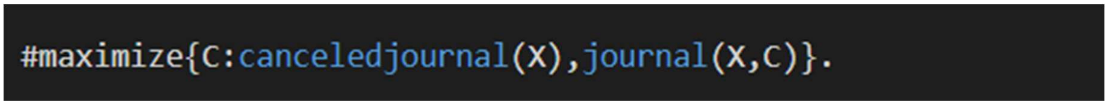
```
:-canceledjournal(X),rating(X,L), 3<=L.
```

The fourth test rule states the journal's total usage. Hence, the aim is to apply the #sum aggregate for performing the total of all the usage and eliminate all the sets that are greater than or equal to p (journal usage) from the canceledjournal.



```
  :-#sum {A,X:canceledjournal(X),usage(X,A)} < p.
```

Last but not least, #maximize is performed on the remaining set of canceledjournal so that the value obtained will contain the highest. Maximize will consider the value of all the costs (C) and based on that it will provide the highest value in the canceledjournal.



```
#maximize{C:canceledjournal(X),journal(X,C)}.
```

2].

Advantages:-

1. **Compact and time-efficient code:-** In logic programming, the developer has to only focus on describing the problem, making the code very compact and time-efficient to write. In ASP the problem-solving part is dived into two components knowledge and solver. The problem here is represented as knowledge and the solver is just like a computer used to address subclasses of problems. In imperative programming languages, the programmer has to follow a step-by-step algorithmic approach to solve a problem and the main aim is "how a problem could be solved?" whereas in logic programming the main focus is on "what the problem is?".

2. **Low maintenance cost of the code:-** Code written in logic programming is easy to read and understand due to which fewer men power would be required to maintain the code, eventually the maintenance cost of the code can be drastically reduced. For example, the code written for the problem mentioned in the coursework is simple and understandable. Moreover, if any check (test) constraints need to be changed then there would not be many complexities hence less time and human power would be required.

Disadvantages:-

1. **Semantic errors are hard to catch:-** While debugging the code in logical programming in terms of syntactic error is easy. However, when there is a semantic fault (logical error) it is very difficult to detect as sometimes the output might not be even generated. For example in ASP, the only hint the user is provided with is "UNSATISFIABLE" which makes the debugging very hard.

2. **Learning logical programming is difficult to learn:-** Firstly, learning logic programming is difficult as the rules are expressed in unique ways (in the form of clauses) that are different from other programming paradigms. Also, rather than using verbose expressions which are more understandable to the users, it uses symbols for defining the rules like "if" is determined by ":-" which might be confusing. Secondly, there are not enough sources on the internet from where this could be learned.