ALC Tableau Algorithm Implementation

V S S Anirudh Sharma and Shreyas Anand Kulkarni

Description of Approach

- There are 3 xml files that are needed for the code:
  - MAN-P7-Input-KB.xml: It contains the ALC Knowledge Base in NNF
  - MAN-P7-Input-Query.xml: It contains the query which we need to check
  - MAN-P6-KB.xml: As discussed during the meeting, we have considered this to add remaining ABox information.
- Firstly, we convert all the xml files to a dictionary in python. We then convert it into a list format for easier manipulation in code
- We convert the negation of the query to NNF format.
- ABox is generated by using the query and Relations and Concepts from MAN-P6-KB.xml

Conventions used

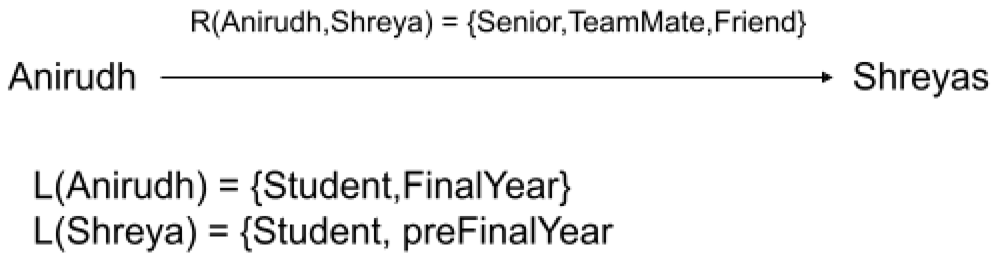
```
#Tbox encoding

* ! = Not
* | = Or
* & = And
* \* = there exists/ some in
* ? = for all/ only in
```

For example:
A and not B would be written as: [‘&’,’A’,[‘!’,’B]]
Related by R with some objects from C : [‘*’,’R’,’C’]
All related by R are from C : [‘?’,’R’,’C’]

The ALC Tableau

THe idea is to uses expansion rules to construct a directed labeled graph (completion graph) $G = \ < V, E, L >$, where V is a set of constants/variables, E is a set of role assertions, and L is a set of node and edge labels L(x) lists the concepts in which x is a member R(x, y) lists the roles in which (x, y) is a member


R(Anirudh,Shreya) = {Senior,TeamMate,Friend}
Anirudh
Shreyas
L(Anirudh) = {Student,FinalYear}
L(Shreya) = {Student, preFinalYear

When a Tableau returns a clash free completion graph then the input sentences are satisfiable (consistent), and that graph represents a model (or family of models) When a Tableau fails to return a clash free completion graph then the input sentences are unsatisfiable (inconsistent) A clash-free completion graph is a finite representation of (infinitely) many models

AND rule

```
if Tnow[0] == '&':
      print('AND RULE:',Tnow[0])
      Tnew = Tnow[1:]+Tbox[1:]+[Tbox[0]]
      return evaluate(L,R,Tnew)
```

OR rule

```
elif Tnow[0] == '|':
      print('OR RULE:',Tnow[0])
      Tnew1 = [Tnow[1]]+Tbox[1:]+[Tbox[0]]
      Tnew2 = [Tnow[2]]+Tbox[1:]+[Tbox[0]]
      return evaluate(L,R,Tnew1) or evaluate(L,R,Tnew2)
```

THERE EXISTS RULE

```
    elif Tnow[0] == '*':
      print('THERE EXISTS RULE:',Tnow[0])
      Lnew = L.copy()
      Rnew = R.copy()
      rel = Tnow[1]
      cl = Tnow[2]

      for a in L:
        Lnew[a].append(Tnow)
        for item in L:
          if not ((a+','+item) in R and rel in R[a+','+item] and cl in L[item]):
            Lnew['var'+a] = [cl]
            Rnew[a+','+'var'+a] = [rel]
      return evaluate(Lnew,Rnew,Tbox[1:])
```

FORALL RULE

```
    elif Tnow[0] == '?':
      print('FOR ALL RULE:',Tnow[0])
      Lnew = L.copy()
      Rnew = R.copy()
      rel = Tnow[1]
      cl = Tnow[2]

      for a in Lnew:
        Lnew[a].append(Tnow)
        for item in L:
          if a+','+item in R and rel in R[a+','+item] and cl not in L[item]:
            Lnew[item].append(cl)
      return evaluate(Lnew,Rnew,Tbox[1:]+[Tbox[0]])
```

BLOCKING RULE

Tableau applies all TBox axioms to all individuals/variables If a rule induces a new variable then applying the same rule to the new variable may induce yet another new variable and so on and Tableau may not terminate We can force a Tableau to terminate if we block the variables from creating new variables that are similar looking to existing variables

Function Description

1. notof(X): Will return not of basic class predicates X
2. abox2L(Rel,Class): Adds data from A-box(Rel and Class) to L(Nodes) and R(Edges)
3. add(Query,L,R): Adds not of Query to our KB (Nodes L and edges R)
4. isConsistent(L): checks if the given graph nodes (L)
5. evaluate(L,R,Tbox): Checks if the given knowledge base has a model or not by applying ALC tableau algorithm
6. printEntailment(L,R,Tbox): Prints whether the Query is entailed by the KB or not.

Limitations

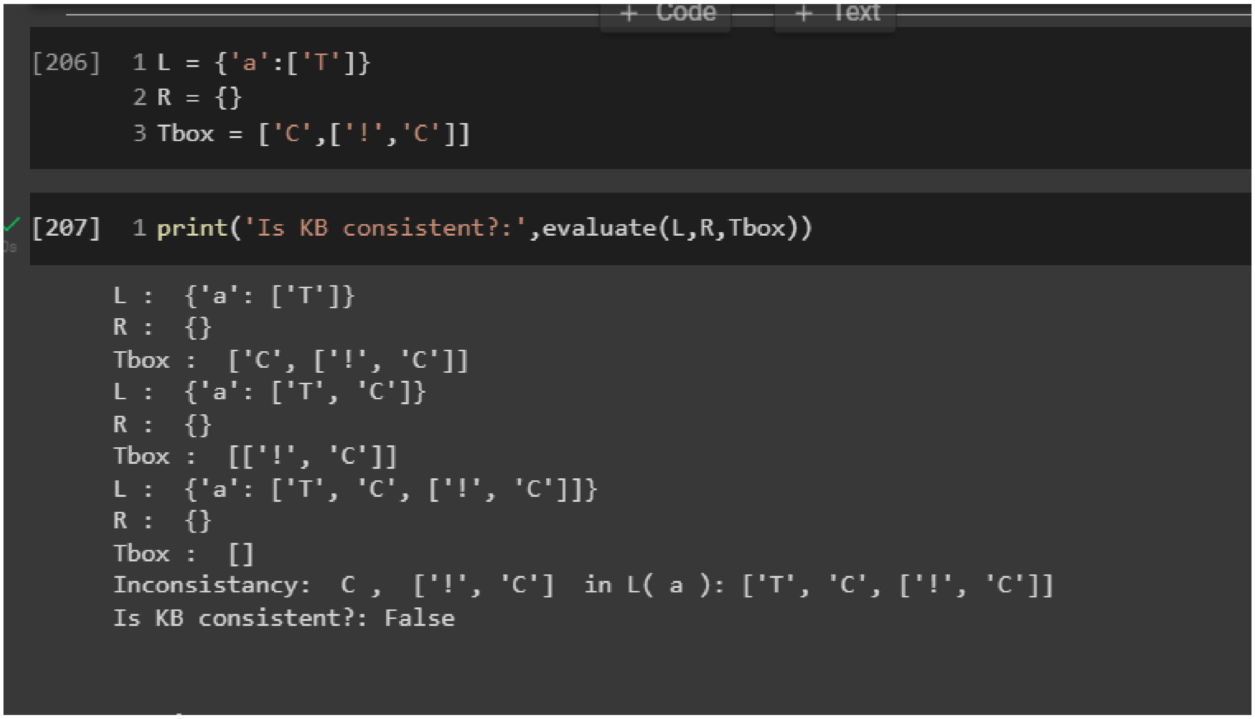
- Will only work for simple cases in forall and there exists operators
  - $\forall R. C$ will work
  - $\forall R. \neg C$ will work
  - But $\forall R. (C \cup D)$ will not work
- Only simple queries of the following formats are allowed:
  - $C(a)$
  - $\neg C(a)$
  - $\exists R. C(a)$
  - $\exists R. \neg C(a)$
  - $\forall R. C(a)$
  - $\forall R. \neg C(a)$

File Details

- main.py : The main code to run the file. **Note: It also contains more samples in addition to the given Sample I/O.**
- README.md : Required README file on how to run the code and the dependencies
- CS6770:Team_7_final_project.pdf

Example Runs

Example 1: Checking system consistency

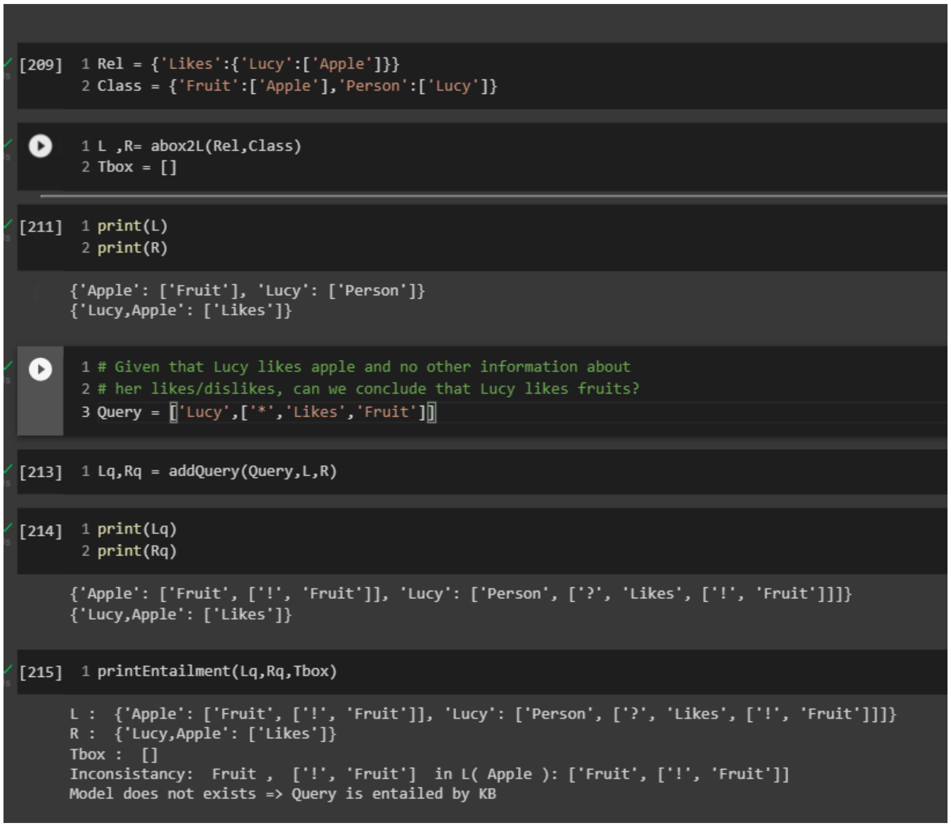


```
L = {'a':['T']}
R = {}
Tbox = ['C',['!','C']]
```

```
print('Is KB consistent?:',evaluate(L,R,Tbox))
```

```
L :  {'a': ['T']}
R :  {}
Tbox :  ['C', ['!', 'C']]
L :  {'a': ['T', 'C']}
R :  {}
Tbox :  [['!', 'C']]
L :  {'a': ['T', 'C', ['!', 'C']]}
R :  {}
Tbox :  []
Inconsistancy:  C ,  ['!', 'C']  in L( a ): ['T', 'C', ['!', 'C']]
Is KB consistent?: False
```

Example 2: Lucy and apples



```
[209] Rel = {'Likes':{'Lucy':['Apple']}}
      Class = {'Fruit':['Apple'],'Person':['Lucy']}
```

```
L ,R= abox2L(Rel,Class)
Tbox = []
```

```
[211] print(L)
      print(R)
```

```
{'Apple': ['Fruit'], 'Lucy': ['Person']}
{'Lucy,Apple': ['Likes']}
```

```
# Given that Lucy likes apple and no other information about
# her likes/dislikes, can we conclude that Lucy likes fruits?
Query = ['Lucy',['*','Likes','Fruit']]
```

```
[213] Lq,Rq = addQuery(Query,L,R)
```

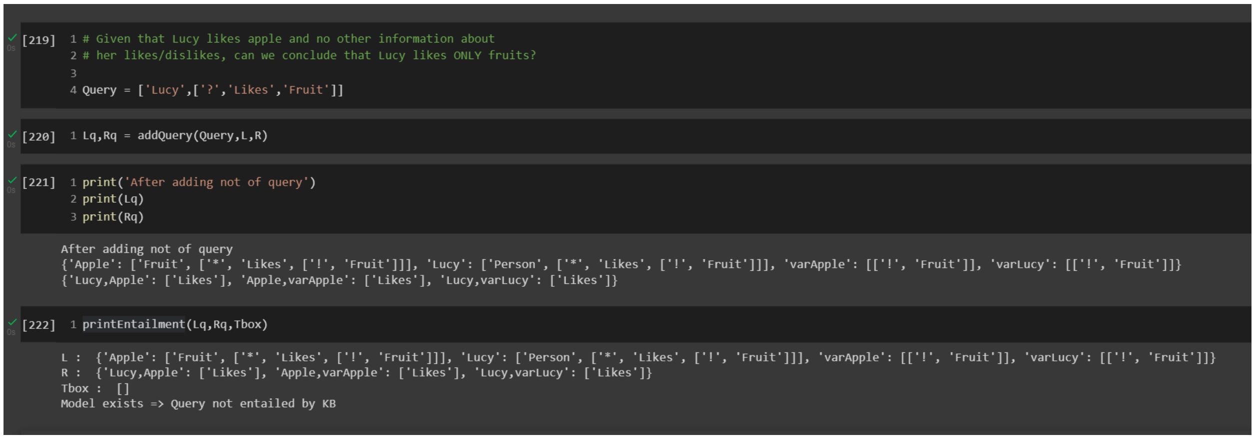
```
[214] print(Lq)
      print(Rq)
```

```
{'Apple': ['Fruit', ['!', 'Fruit']], 'Lucy': ['Person', ['?', 'Likes', ['!', 'Fruit']]]}
{'Lucy,Apple': ['Likes']}
```

```
[215] printEntailment(Lq,Rq,Tbox)
```

```
L :  {'Apple': ['Fruit', ['!', 'Fruit']], 'Lucy': ['Person', ['?', 'Likes', ['!', 'Fruit']]]}
R :  {'Lucy,Apple': ['Likes']}
Tbox :  []
Inconsistancy:  Fruit ,  ['!', 'Fruit']  in L( Apple ): ['Fruit', ['!', 'Fruit']]
Model does not exists => Query is entailed by KB
```

Example 3: Lucy and more apples



```
[219] # Given that Lucy likes apple and no other information about
      # her likes/dislikes, can we conclude that Lucy likes ONLY fruits?

      Query = ['Lucy',['?','Likes','Fruit']]
```

```
[220] Lq,Rq = addQuery(Query,L,R)
```

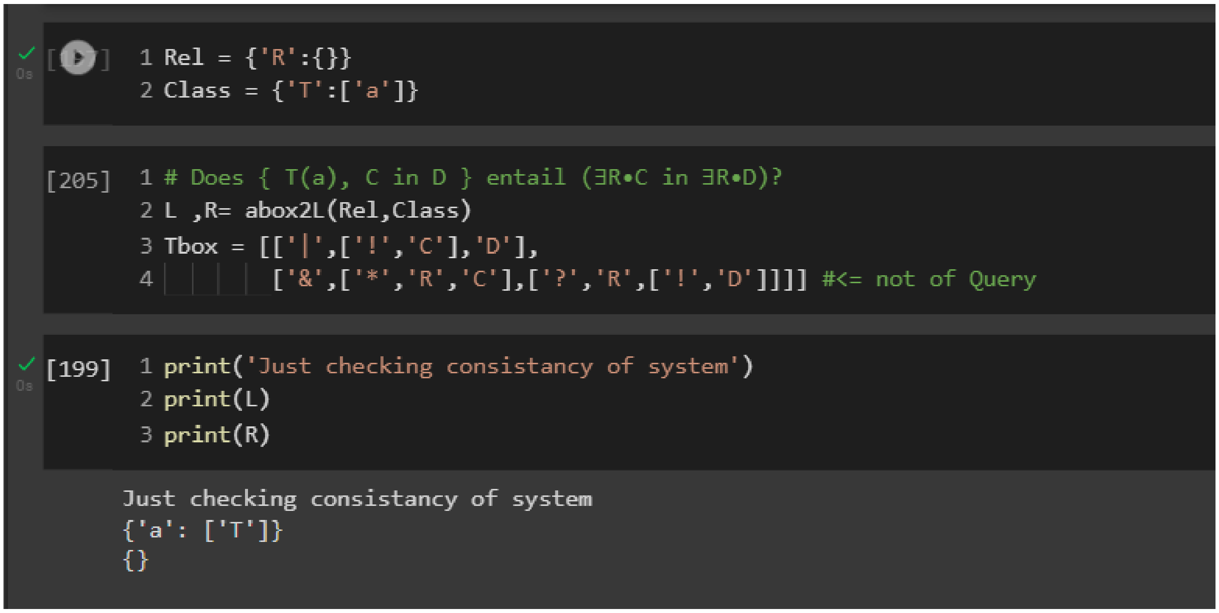
```
[221] print('After adding not of query')
      print(Lq)
      print(Rq)
```

```
After adding not of query
{'Apple': ['Fruit', ['*', 'Likes', ['!', 'Fruit']]], 'Lucy': ['Person', ['*', 'Likes', ['!', 'Fruit']]], 'varApple': [['!', 'Fruit']], 'varLucy': [['!', 'Fruit']]}
{'Lucy,Apple': ['Likes'], 'Apple,varApple': ['Likes'], 'Lucy,varLucy': ['Likes']}
```

```
[222] printEntailment(Lq,Rq,Tbox)
```

```
L :  {'Apple': ['Fruit', ['*', 'Likes', ['!', 'Fruit']]], 'Lucy': ['Person', ['*', 'Likes', ['!', 'Fruit']]], 'varApple': [['!', 'Fruit']], 'varLucy': [['!', 'Fruit']]}
R :  {'Lucy,Apple': ['Likes'], 'Apple,varApple': ['Likes'], 'Lucy,varLucy': ['Likes']}
Tbox :  []
Model exists => Query not entailed by KB
```

Example 4:

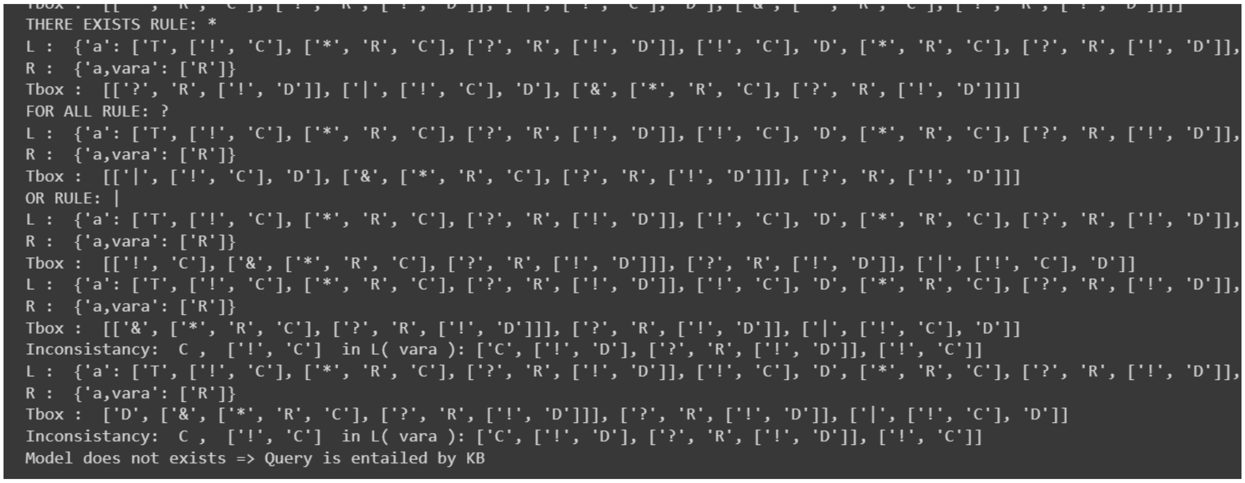


```
Rel = {'R':{}}
Class = {'T':['a']}
```

```
# Does { T(a), C in D } entail (∃R•C in ∃R•D)?
L ,R= abox2L(Rel,Class)
Tbox = [['|',['!','C'],'D'],
        ['&',['*','R','C'],['?','R',['!','D']]]] #<= not of Query
```

```
print('Just checking consistancy of system')
print(L)
print(R)
```

```
Just checking consistancy of system
{'a': ['T']}
{}
```



```
THERE EXISTS RULE: *
L :  {'a': ['T', ['!', 'C'], ['*', 'R', 'C'], ['?', 'R', ['!', 'D']], ['!', 'C'], 'D', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']],
R :  {'a,vara': ['R']}
Tbox :  [['?', 'R', ['!', 'D']], ['|', ['!', 'C'], 'D'], ['&', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']]]]
FOR ALL RULE: ?
L :  {'a': ['T', ['!', 'C'], ['*', 'R', 'C'], ['?', 'R', ['!', 'D']], ['!', 'C'], 'D', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']],
R :  {'a,vara': ['R']}
Tbox :  [['|', ['!', 'C'], 'D'], ['&', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']]], ['?', 'R', ['!', 'D']]]
OR RULE: |
L :  {'a': ['T', ['!', 'C'], ['*', 'R', 'C'], ['?', 'R', ['!', 'D']], ['!', 'C'], 'D', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']],
R :  {'a,vara': ['R']}
Tbox :  [['!', 'C'], ['&', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']]], ['?', 'R', ['!', 'D']], ['|', ['!', 'C'], 'D']]
L :  {'a': ['T', ['!', 'C'], ['*', 'R', 'C'], ['?', 'R', ['!', 'D']], ['!', 'C'], 'D', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']],
R :  {'a,vara': ['R']}
Tbox :  [['&', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']]], ['?', 'R', ['!', 'D']], ['|', ['!', 'C'], 'D']]
Inconsistancy:  C ,  ['!', 'C']  in L( vara ): ['C', ['!', 'D'], ['?', 'R', ['!', 'D']], ['!', 'C']]
L :  {'a': ['T', ['!', 'C'], ['*', 'R', 'C'], ['?', 'R', ['!', 'D']], ['!', 'C'], 'D', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']],
R :  {'a,vara': ['R']}
Tbox :  ['D', ['&', ['*', 'R', 'C'], ['?', 'R', ['!', 'D']]], ['?', 'R', ['!', 'D']], ['|', ['!', 'C'], 'D']]
Inconsistancy:  C ,  ['!', 'C']  in L( vara ): ['C', ['!', 'D'], ['?', 'R', ['!', 'D']], ['!', 'C']]
Model does not exists => Query is entailed by KB
```