Artificial Intelligence Assignment 4 Readme Document

Author:
Joshua Wang
UNI: jw3123

1. The programming language in the project is Python
2. The version of programming language is python 2.7 (64bits)
3. Development environment is:
   Application: Sublime text2
   Operating System: Windows 8
   Processor: Intel i7-4702MQ
   Memory: 8GB DDR3
4. How run on CLIC:

Type:

(chmod 755 entail)

./entail <algorithm type> <KB file> <query>

Where <algorithm type> supports “forward”, “backward”, “resolution”.

If the query is true, the solution will be printed

5. Design of the programs:
   A. Forward Chaining:
      Overall algorithm:
         Deduct the number of premises of each rule when the symbol is known to be true. When the number becomes 0, check whether the conclusion is the query (or “_NULL_”). If not, put the conclusion to the pool of symbols known to be true.

      Special case:
      1. Query is negative: check whether there is a corresponding negative fact.
      2. KB is not consistent: each clause of the form of all negative literal (~A v ~B v ~C) will be transformed into (A ^ B ^ C => “_NULL_”). Thus when “_NULL_” is deduced, we known there is a contradiction in the KB. When this happen, the conclusion is true.

   B. Backward Chaining:
      Overall algorithm:
         Recursively call the function to check whether all premises are true.

Special case:

1. Query is negative: check whether there is a corresponding negative fact.
2. KB is not consistent: before query check whether KB entail "_NULL_" (principle is the same as that in that in forward algorithm)
3. Loop: retain a set seen recording the symbols seen in the upper levels of recursion. If a loop is detect, there is an assignment that makes the query false while KB true. I.e., all symbols in the loop false. Thus this path fails.

C. Resolution:

Overall algorithm:

For each pair of clauses in the KBset (originally KB ^ ~query, but will expand), check whether there is an empty clause in all possible resolvants. If there is, output true. Else add the resolvants to the set "new". If KBset no longer expands ("new" is a subset of KBset), return false.

Print out the result:

Keep a backpointer, recording the origin of each clause in KBset. Since the clause might be generated multiple times, use a stack to store all of them. If the query is entailed, reclusively print the clauses, from rules in KB in the beginning, to "NULL" in the end.

6. Examples:

./entail forward for_1.txt queen

for_1.txt: # a variation of the example on the website by mixing forms of horn form and transferring to strings as variables

pig => queen # if pig then queen
lion ^ man => pig # if lion and man then pig
boy ^ lion => man # if boy and lion then man
~apple v ~pig v lion # not apple or not pig or lion (is true)
~apple v lion v ~boy # not apple or lion or not boy (is true)
apple #apple is true
boy #boy is true

Output:
apple

boy
~apple v lion v ~boy
boy ^ lion => man
lion ^ man => pig
pig => queen
--> true

./entail forward for_2.txt egg # a loop that egg implies fish, and fish implies egg

apple ^ book ^ fish => cat # apple and book and fish implies cat
cat ^ dog => egg # cat and dog implies egg
egg => fish # egg implies fish
apple # apple is true
book #book is true
dog # dog is true

Output:
--> false

./entail backward back_1.txt egg
back_1.txt: # cat implies dog, cat and dog implies egg
apple #apple is true
boy # boy is true
apple ^ boy => cat # apple and boy implies cat
~cat v dog # not cat or dog is true
cat ^ dog => egg # cat and dog implies egg

Output:
cat ^ dog => egg
apple ^ boy => cat
apple
boy
~cat v dog
--> true

./entail backward back_2.txt egg
back_2.txt: # a loop from dog to boy, and boy to dog
apple #apple is true
dog => boy # dog implies boy
apple ^ boy => cat # apple and boy implies cat
~cat v dog # not cat or dog is true
cat ^ dog => egg # cat and dog implies egg

output:
--> false

./entail CNF resol_1.txt hasgreen
resol_1.txt: # the object that hasblue and thus must hasred, hasgreen, but not necessarily has black
(hasred v hasgreen v hasblue v hasblack)
(the object) either has red, green, blue, black colors
(~hasred v hasgreen) # Either not red, or green or both
(~hasblue v hasred) # Either not blue or red or both
hasblue # has blue color

Output:
resolve: (~hasred^ hasgreen) and (~hasgreen) --> (~hasred)
resolve: (~hasblue^ hasred) and (hasblue) --> (hasred)
resolve: (~hasred) and (hasred) --> NULL
--> true

./entail CNF resol_2.txt A
resol_2.txt:
(~B v P v M) ^ (B v ~P) ^ (B v ~M) ^ (A v B v C)
(not B or P or M) and (B or not P) and (B or not M) and (A or B or C) is true
Output:
--> false

./entail CNF resol_3.txt hasgreen

resol_3.txt: # the object that hasblue and thus must hasred, but not necessarily has black and green

(hasred v hasgreen v hasblue v hasblack)
(hasred v hasgreen)
(~hasblue v hasred)
hasblue

Output:
--> false

./entail CNF resol_4.txt hasblack

resol_4.txt: # the object that has all colors
(hasred v hasgreen v hasblue v hasblack) ^ (~hasred v hasgreen) ^ (~hasblue v hasred) ^ (~hasblue v hasblack v ~hasred) ^ hasblue

Outputs:
resolve: (~hasblue^ hasred) and (hasblue) --> (hasred)
resolve: (~hasblue^ hasblack^ ~hasred) and (~hasblack) --> (~hasblue^ ~hasred)
resolve: (~hasblue^ ~hasred) and (hasblue) --> (~hasred)
resolve: (hasred) and (~hasred) --> NULL
--> true

./entail CNF resol_5.txt hasblack

resol_5.txt: # the object that only has black

(hasred v hasgreen v hasblue v hasblack)
(hasred v ~hasgreen)
(hasblue v ~hasred)
~hasblue

Outputs:

resolve: (~hasblue) and (hasred^ hasgreen^ hasblue^ hasblack) --> (hasred^ hasgreen^ hasblack)
resolve: (hasred^ hasgreen^ hasblack) and (~hasblack) --> (hasred^ hasgreen)
resolve: (hasblue^ ~hasred) and (hasred^ ~hasgreen) --> (hasblue^ ~hasgreen)
resolve: (hasred^ hasgreen) and (hasblue^ ~hasgreen) --> (hasred^ hasblue)
resolve: (hasblue^ ~hasred) and (hasred^ hasblue) --> (hasblue)
resolve: (~hasblue) and (hasblue) --> NULL
--> true

KB inconsistent
./entail backward inconsistentKB.txt Joshua

inconsistentKB.txt: # the KB that is inconsistent
~apple v book # not apple or book
apple # apple is true
~book # not book is true

Output:
Note: KB is not consistent, thus it entails anything
~book
~apple v book
apple
--> true