ReadMe

-----ns2786 Nianwen Song

1. Heuristic Function

I use unit-preference heuristic function in
the program. This function will add one unit cost to each clause in the knowledge base
except the unit clauses (I the just choose node-depth attribute to stored the picked clause
in each step). So the unit clauses will have a higher priority and the
program will select them first.

2. Pruning Strategies

I uses two pruning strategies in the search procedure. First, it will cut off the
branch when the two clauses cannot
resolution. Another strategy is to prune when the length of resolvent of two clauses is
equal to the sum of length of them. In this case, the resolution procedure is just combine
two clauses and will not simplify the problem.

3. Test Examples

Test1
The knowledge base and the query is the scuba diver example in ppt.

```
(defparameter kb '(_AND

            (_OR (WS ?x) (SD ?x))

            (_OR (_NOT (SD ?y)) (_NOT (Like ?y Waves)))

            (_OR (_NOT (WS ?z)) (Like ?z Warm))

            (_OR (_NOT (Like Laura ?w)) (_NOT (Like Jacob ?w)))

            (_OR (Like Jacob ?w) (Like Laura ?w))

            (Like Jacob Warm)

            (Like Jacob Waves)))



(defparameter nq '(_AND (_OR (_NOT (SD ?V) (WS ?V)))))
```

Output result:

```
CL-USER> (atp kb nq)
((((_NOT (SD ?V) (WS ?V))) ((WS ?X) (SD ?X)) (WS ?X) ((?V . ?X)))
 (((WS ?X)) ((_NOT (WS ?Z)) (LIKE ?Z WARM)) (LIKE ?Z WARM) ((?X . ?Z)))
 (((LIKE ?Z WARM)) ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_NOT (LIKE JACOB WARM)) ((?W . WARM) (?Z . LAURA)))
 (((_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM)) NIL (NIL)))
```

Explanation :

```
(
        (
         ((_NOT (SD ?V) (WS ?V)))  ;No scuba diver likes big waves, and all water skiers like
warm water.
         ((WS ?X) (SD ?X)) ;Every member of the CRC is either a scuba diver or a water skier
or both
         (WS ?X)   ;resolution
         ((?V . ?X)) ;theta
         )

        (
         ((WS ?X))
         ((_NOT (WS ?Z)) (LIKE ?Z WARM))  ;Laura dislikes whatever Jacob likes, and likes
whatever Jacob dislikes.
         (LIKE ?Z WARM)
         ((?X . ?Z))
         )

          (
         ((LIKE ?Z WARM))
         ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
            (_NOT (LIKE JACOB WARM))
         ((?W . WARM) (?Z . LAURA))
         )

          (
         ((_NOT (LIKE JACOB WARM)))
         ((LIKE JACOB WARM))
         NIL    ;reach the goal
         (NIL)
         )
        )
```

Test 2(test skolem function)
The knowledge base and the query is the customs officials example in ppt.

(defparameter kb1

'(_AND (_OR (_NOT (E ?x)) (V ?x) (S ?x (@f ?x)))

```
            (P c)

            (_OR (_NOT (E ?x)) (V ?x) (C (@f ?x)))

            (E ?c)

            (_OR (_NOT (S C ?y)) (P ?y))

            (_OR (_NOT (P ?z)) (_NOT (V ?z)))))
```

```
(defparameter nq1

        '(_AND (_OR (_NOT (P ?w)) (_NOT (C ?w)))))
```

Output result:

```
CL-USER> (atp kb1 nq1)
((((_NOT (P ?W)) (_NOT (C ?W))) ((P C)) (_NOT (C C)) ((?W . C)))
 (((_NOT (C C))) ((_NOT (E ?X)) (V ?X) (C (@F ?X))) (_OR (_NOT (E ?X)) (V ?X))
  (((@F ?X) . C)))
 (((_NOT (E ?X)) (V ?X)) ((E ?C)) (V ?X) ((?C . ?X)))
 (((V ?X)) ((_NOT (P ?Z)) (_NOT (V ?Z))) (_NOT (P ?Z)) ((?X . ?Z)))
 (((_NOT (P ?Z))) ((P C)) NIL ((?Z . C))))
```

Explanation:

```
(
       (
        ((_NOT (P ?W)) (_NOT (C ?W)))
        ((P C))              ;unit-preference heuristic
        (_NOT (C C))
        ((?W . C))
        )

        (
         ((_NOT (C C)))
         ((_NOT (E ?X)) (V ?X) (C (@F ?X)))
         (_OR (_NOT (E ?X)) (V ?X))
         (((@F ?X) . C))  ;binding of skolem function
         )

        (
         ((_NOT (E ?X)) (V ?X))
         ((E ?C))
         (V ?X)
         ((?C . ?X))
```

```
          )

        (
            ((V ?X))
            ((_NOT (P ?Z)) (_NOT (V ?Z)))
            (_NOT (P ?Z))
            ((?X . ?Z))
         )

         (
          ((_NOT (P ?Z)))
          ((P C))
          NIL
          ((?Z . C))
          )
```

Test 3(test function)
KB is
John is a columbia student.
All columbia students take at least one course

Query is
Does John take a course?

Output result:

```
CL-USER> (atp '(_AND
  (_OR
      (_NOT (Columbia ?x))
      (isCourse (@sf ?x))
  )
  (_OR
      (_NOT (Columbia ?x))
      (TakeCourse ?x (@sf ?x))
  )
  (_OR
      (Columbia John)
  )
)

'(_AND
 (_OR
     (_NOT (isCourse ?y))
     (_NOT (TakeCourse John ?y))
 ))
)
```

NIL ; means query(~nq) is false based on the KB, so John takes a course.

Test 3(complicated example)

It is the American criminal example in professor's PPT
KB is:

```
(defparameter kb4
        '(_AND (_OR (_NOT (American ?x)) (_NOT (Weapon ?y)) (_NOT (Sells ?x ?y ?
z)) (_NOT (Hostile ?z))(Criminal ?x))
          (Owns Foo M1)
          (Missle M1)
          (_OR (_NOT (Missle ?x)) (_NOT (Owns Foo ?x)) (Sells West ?x Foo))
          (_OR (_NOT (Missle ?x)) (Weapon ?x))
          (_OR (_NOT (Enemy ?x America)) (Hostile ?x))
          (American West)
          (Enemy Foo America)))
```

Query is :

```
(defparameter nq4 '(_AND '(_NOT (Criminal West))))
```

Output result:

```
CL-USER> (atp kb4 nq4)
((((_NOT (CRIMINAL WEST)))  ((_NOT (AMERICAN ?X)) (_NOT (WEAPON ?Y)) (_NOT
(SELLS ?X ?Y ?Z))  (_NOT (HOSTILE ?Z)) (CRIMINAL ?X))(_OR (_NOT (AMERICAN
WEST)) (_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y ?Z)) (_NOT (HOSTILE ?Z)))  ((?X .
WEST)))

(((_NOT (AMERICAN WEST)) (_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y ?Z))(_NOT
(HOSTILE ?Z))) ((AMERICAN WEST)) (_OR (_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y
?Z)) (_NOT (HOSTILE ?Z))) (NIL))

 (((_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y ?Z)) (_NOT (HOSTILE ?Z))) ((_NOT
(ENEMY ?X AMERICA)) (HOSTILE ?X))  (_OR (_NOT (WEAPON ?Y)) (_NOT (SELLS
WEST ?Y ?X)) (_NOT (ENEMY ?X AMERICA)))  ((?Z . ?X)))

 (((_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y ?X)) (_NOT (ENEMY ?X AMERICA)))
  ((ENEMY FOO AMERICA)) (_OR (_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y FOO)))
  ((?X . FOO)))

 (((_NOT (WEAPON ?Y)) (_NOT (SELLS WEST ?Y FOO)))((_NOT (MISSLE ?X)) (WEAPON ?
X)) (_OR (_NOT (SELLS WEST ?X FOO)) (_NOT (MISSLE ?X))) ((?Y . ?X)))

 (((_NOT (SELLS WEST ?X FOO)) (_NOT (MISSLE ?X))) ((MISSLE M1)) (_NOT (SELLS
WEST M1 FOO)) ((?X . M1)))

 (((_NOT (SELLS WEST M1 FOO))) ((_NOT (MISSLE ?X)) (_NOT (OWNS FOO ?X)) (SELLS
WEST ?X FOO)) (_OR (_NOT (MISSLE M1)) (_NOT (OWNS FOO M1))) ((?X . M1)))

 (((_NOT (MISSLE M1)) (_NOT (OWNS FOO M1))) ((MISSLE M1)) (_NOT (OWNS FOO
M1))  (NIL))

 (((_NOT (OWNS FOO M1))) ((OWNS FOO M1)) NIL (NIL)))
```

Use 9 steps to reach the result

4. Extra Points

The program converts FOL to SNF in the following order. First, it will eliminate the implication in the FOL. Second, it will move outside '_NOT in and duplicate '_NOT will be removed. Third, standardize variables. Forth, skolemize. Fifth, the program will extract all qualifiers and drop them.Finally, distributing. At first, I didn't consider the standized variables step, so I do it later, due to the time constraint, I finish these two part separately, but did not combine them. Since after that, we get an deadline extension, so I combine them together at last.

5. Test Examples

Standarized Variable

case1:

```
CL-USER> (stand-var '(_ALL ?z(_OR (WS ?z) (SD ?z))))

case1 output:
(_ALL ?X (_OR (WS ?X) (SD ?X)))     ;the assigned variables changes, I pick them
                                            ;sequentially from a created list(?x ?x ?y ?y ...etc)
```

case2:

```
CL-USER> (stand-var '(_OR (_ALL ?x (_EXI ?y (_AND (Test1 ?y) (_NOT (Test2 ?x ?y)))))
(_EXI ?y (Test3 ?y ))))

case2 output:
(_OR (_ALL ?A (_EXI ?B (_AND (TEST1 ?B) (_NOT (TEST2 ?A ?B)))))
 (_EXI ?C (TEST3 ?C)))
```

FOL--->SNF

case1:

```
CL-USER> (fol '(_AND
                    (_ALL ?x (_OR (WS ?x) (SD ?x)))
                    (_NOT (_EXI ?y (_AND (SD ?y) (Like ?y Waves))))
                    (_ALL ?z (=> (WS ?z) (Like ?z Warm)))
                    (_ALL ?w (<=> (Like Laura ?w) (_NOT (Like Jacob ?w))))
                    (Like Jacob Warm)
                    (Like Jacob Waves)))
```

case1 output:

```
(_AND (_OR (WS ?X) (SD ?X)) (_OR (_NOT (SD ?Y)) (_NOT (LIKE ?Y WAVES)))
 (_OR (_NOT (WS ?Z)) (LIKE ?Z WARM))
 (_OR (_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
 (_OR (LIKE LAURA ?W) (LIKE JACOB ?W)) (LIKE JACOB WARM) (LIKE JACOB WAVES))
```

case2(No redudant '_AND):

```
CL-USER> (fol '(_NOT (_EXI ?v (_AND (SD ?v) (_NOT (WS ?v))))))
```

case2 output:

```
(_AND (_OR (_NOT (SD ?V)) (WS ?V)))
```

case3(redudant '_AND):

```
CL-USER> (fol '(_AND (_NOT (_EXI ?v (_AND (SD ?v) (_NOT (WS ?v)))))))
```

case3 output:

```
(_AND (_OR (_NOT (SD ?V)) (WS ?V)))
```

FOL--->ATP(without standized variable version)

Test1:

```
CL-USER> (defparameter fkb '(_AND
           (_ALL ?x (_OR (WS ?x) (SD ?x)))
                     (_NOT (_EXI ?y (_AND (SD ?y) (Like ?y Waves))))
                     (_ALL ?z (=> (WS ?z) (Like ?z Warm)))
                     (_ALL ?w (<=> (Like Laura ?w) (_NOT (Like Jacob ?w))))
           (Like Jacob Warm)
           (Like Jacob Waves)))

(defparameter fnq '(
           _NOT (_EXI ?v (_AND (SD ?v) (_NOT (WS ?v))))))
```

Output result(set successor cost=1):

```
CL-USER> (atp-fol fkb fnq)

((((_NOT (SD ?V)) (WS ?V)) ((WS ?X) (SD ?X)) (WS ?X) ((?V . ?X)))
 (((WS ?X)) ((_NOT (WS ?Z)) (LIKE ?Z WARM)) (LIKE ?Z WARM) ((?X . ?Z)))
 (((LIKE ?Z WARM)) ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_NOT (LIKE JACOB WARM)) ((?W . WARM) (?Z . LAURA)))
 (((_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM)) NIL (NIL)))
```

Output result(set successor cost=0):

```
CL-USER> (atp-fol fkb fnq)
((((_NOT (SD ?V)) (WS ?V)) ((_NOT (WS ?Z)) (LIKE ?Z WARM))
  (_OR (_NOT (SD ?Z)) (LIKE ?Z WARM)) ((?V . ?Z)))
 (((_NOT (SD ?Z)) (LIKE ?Z WARM))
  ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_OR (_NOT (SD LAURA)) (_NOT (LIKE JACOB WARM))) ((?W . WARM) (?Z . LAURA)))
 (((_NOT (SD LAURA)) (_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM))
  (_NOT (SD LAURA)) (NIL))
 (((_NOT (SD LAURA))) ((WS ?X) (SD ?X)) (WS LAURA) ((?X . LAURA)))
 (((WS LAURA)) ((_NOT (WS ?Z)) (LIKE ?Z WARM)) (LIKE LAURA WARM)
  ((?Z . LAURA)))
 (((LIKE LAURA WARM)) ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_NOT (LIKE JACOB WARM)) ((?W . WARM)))
 (((_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM)) NIL (NIL)))
```

Test2(skolemize):

```
CL-USER> (fol '((_ALL ?x (=> (_AND (E ?x) (_NOT (V ?x))) (_EXI ?y (_AND (S ?x ?y) (C ?
y)))))))
```

Output result:

```
(_AND
 ((_OR (_NOT (_AND (E ?X) (_NOT (V ?X)))) (_AND (S ?X (@F ?X)) (C (@F ?X))))))
```

FOL--->ATP(with standized variable version)

Final Test:

Input:

```
CL-USER> (defparameter fkb '(_AND
(_ALL ?x (_OR (WS ?x) (SD ?x)))
(_NOT (_EXI ?x (_AND (SD ?x) (Like ?x Waves))))
(_ALL ?x (=> (WS ?x) (Like ?x Warm)))
(_ALL ?x (<=> (Like Laura ?x) (_NOT (Like Jacob ?x))))
(Like Jacob Warm)
(Like Jacob Waves)))

(defparameter fnq '(
_NOT (_EXI ?x (_AND (SD ?x) (_NOT (WS ?x))))))

(atp-fol fkb fnq)
```

Output:

```
((((_NOT (SD ?V)) (WS ?V)) ((_NOT (WS ?Z)) (LIKE ?Z WARM))
  (_OR (_NOT (SD ?Z)) (LIKE ?Z WARM)) ((?V . ?Z)))
 (((_NOT (SD ?Z)) (LIKE ?Z WARM))
  ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_OR (_NOT (SD LAURA)) (_NOT (LIKE JACOB WARM))) ((?W . WARM) (?Z . LAURA)))
 (((_NOT (SD LAURA)) (_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM))
  (_NOT (SD LAURA)) (NIL))
 (((_NOT (SD LAURA))) ((WS ?X) (SD ?X)) (WS LAURA) ((?X . LAURA)))
 (((WS LAURA)) ((_NOT (WS ?Z)) (LIKE ?Z WARM)) (LIKE LAURA WARM)
  ((?Z . LAURA)))
 (((LIKE LAURA WARM)) ((_NOT (LIKE LAURA ?W)) (_NOT (LIKE JACOB ?W)))
  (_NOT (LIKE JACOB WARM)) ((?W . WARM)))
 (((_NOT (LIKE JACOB WARM))) ((LIKE JACOB WARM)) NIL (NIL)))
```