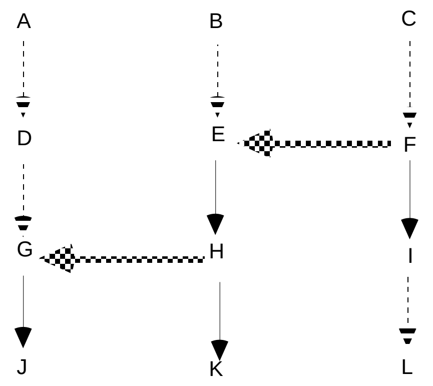
IV
SENTENTIAL REASONING

The general architecture of OSCAR has been described, and default definitions have been provided for such concepts as i-preference. Programming OSCAR consists of supplying arrays of reason schemas, permanent ultimate epistemic interests, setting the value of α_0, and (optionally) replacing some of the default definitions. In applying OSCAR to reasoning problems, we will typically want the system to employ both substantive reasons (which may vary from problem to problem) and logical inference rules (which will tend to remain constant from problem to problem.) I will begin this chapter by discussing OSCAR's performance on some of the problems introduced in chapter three of *Cognitive Carpentry* that do not involve logic. Then I will turn to the implementation of purely deductive reasoning in the propositional calculus. In section three, I will discuss some surprising problems that arise when we combine the logical inference rules of section two with the defeasible reasoning discussed in section one.

1. Reasoning without Logic

OSCAR can be used without logic to deal with many of the reasoning problems that were discussed above and in *Cognitive Carpentry*. This can be illustrated with problems contained in the problem-set *Agent-probs.lsp*. Consider the problem diagrammed in figure 2 of chapter one. This is problem #3 in the problem-set. The following is a display of OSCAR's reasoning, produced by turning on proofs (setting "(proof-on)"), display-mode (setting "(display-on)"), and using no logical inference-rules. The latter is accomplished by executing "(logic-off)".

```
Problem #3
Figure 2
Given premises:
    A    justification = 1
    B    justification = 1
    C    justification = 1
Ultimate epistemic interests:
    J    interest = 1
    K    interest = 1
    L    interest = 1

  FORWARDS PRIMA FACIE REASONS
    pf-reason 1:   {A} ||=> D   strength = 1
    pf-reason 2:   {D} ||=> G   strength = 1
    pf-reason 3:   {B} ||=> E   strength = 1
    pf-reason 4:   {C} ||=> F   strength = 1
    pf-reason 5:   {I} ||=> L   strength = 1

  FORWARDS CONCLUSIVE REASONS
    con-reason 1:   {G} ||=> J   strength = 1
    con-reason 2:   {E} ||=> H   strength = 1
    con-reason 3:   {H} ||=> K   strength = 1
    con-reason 4:   {F} ||=> I   strength = 1
    con-reason 5:   {F} ||=> (B @ E)   strength = 1
    con-reason 6:   {H} ||=> (D @ G)   strength = 1
```



```
 # 1   A
 given
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1


 # 2   B
 given
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1


 # 3   C
 given
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: J> from the inference-queue.
                                  # 1
                                  interest: J
                                  This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Query #2: K> from the inference-queue.
                                  # 2
                                  interest: K
                                  This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Query #3: L> from the inference-queue.
                                  # 3
                                  interest: L
                                  This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.
                                  # 4
                                  interest: (C @ F)
                                  Of interest as defeater for support-link 1
                                  # 5
                                  interest: ~F
                                  Of interest as defeater for support-link 1

 # 4   F
 Inferred by support-link #1 from { 3 } by pf-reason 4
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.

 # 5   I
 Inferred by support-link #2 from { 4 } by con-reason 4
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is not defeasible.

 # 6   (B @ E)
 Inferred by support-link #3 from { 4 } by con-reason 5
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is not defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.
                                  # 6
                                  interest: (I @ L)
                                  Of interest as defeater for support-link 4
                                  # 7
                                  interest: ~L
```

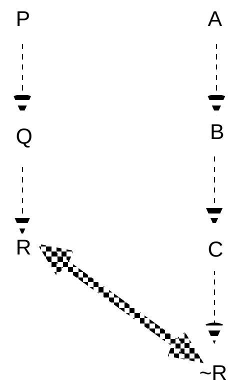
```
                                        Of interest as defeater for support-link 4

 # 7   L
 Inferred by support-link #4 from { 5 } by pf-reason 5
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is defeasible.
        ========================================
        Justified belief in L
        answers #<Query #3: L> affirmatively.
        ========================================
----------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
----------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
                                        # 8
                                        interest: (B @ E)
                                        Of interest as defeater for support-link 5
 Node # 6 defeats link # 5
                                        # 9
                                        interest: ~E
                                        Of interest as defeater for support-link 5

 # 8   E                  DEFEATED
 Inferred by support-link #5 from { 2 } by pf-reason 3  defeaters: { 6 }
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 0
 This inference is defeasible.
----------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.
                                        # 10
                                        interest: (A @ D)
                                        Of interest as defeater for support-link 6
                                        # 11
                                        interest: ~D
                                        Of interest as defeater for support-link 6

 # 9   D
 Inferred by support-link #6 from { 1 } by pf-reason 1
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is defeasible.
----------------------------------------------------------------------------
Retrieving #<Node 9> from the inference-queue.
                                        # 12
                                        interest: (D @ G)
                                        Of interest as defeater for support-link 7
                                        # 13
                                        interest: ~G
                                        Of interest as defeater for support-link 7

 # 10   G
 Inferred by support-link #7 from { 9 } by pf-reason 2
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is defeasible.
----------------------------------------------------------------------------
Retrieving #<Node 10> from the inference-queue.

 # 11   J
 Inferred by support-link #8 from { 10 } by con-reason 1
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is not defeasible.
        ========================================
        Justified belief in J
        answers #<Query #1: J> affirmatively.
```

```
               ========================================
---------------------------------------------------------------------------
Retrieving #<Node 11> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 8> from the inference-queue.

  # 12   H                  DEFEATED
  Inferred by support-link #9 from { 8 } by con-reason 2
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is not defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 12> from the inference-queue.

  # 13   K                  DEFEATED
  Inferred by support-link #10 from { 12 } by con-reason 3
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is not defeasible.
               ----------------------------------------
               #<Node 13> answers #<Query #2: K>
               ----------------------------------------
  Node # 14 defeats link # 7
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 11> . 1) (#<Node 10> . 1) (#<Node 14> . 0) (#<Node 13> . 0) (#<Node 12> . 0)
.   (#<Node 9> . 1) (#<Node 8> . 0) (#<Node 7> . 1) (#<Node 6> . 1) (#<Node 5> . 1)
.   (#<Node 4> . 1) (#<Node 3> . 1) (#<Node 2> . 1) (#<Node 1> . 1))

  # 14   (D @ G)                  DEFEATED
  Inferred by support-link #11 from { 12 } by con-reason 6
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is not defeasible.
  defeatees: { 7 }
---------------------------------------------------------------------------
Retrieving #<Node 13> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 13: ~G supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 11: ~D supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 9: ~E supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 7: ~L supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 5: ~F supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 14> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 12: (D @ G) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 10: (A @ D) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 8: (B @ E) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 6: (I @ L) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 4: (C @ F) supposing {  }> from the inference-queue.


================= ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in L
  is answered affirmatively by node 7
```

```
----------------------------------------------------
  Interest in K
  is unsatisfied.
----------------------------------------------------
  Interest in J
  is answered affirmatively by node 11
----------------------------------------------------

Elapsed time = 5.77 sec

================================================================================
ARGUMENT #1
This is an undefeated argument of strength 1 for:
     L
 which is of ultimate interest.

 3.  C     given
 4.  F     pf-reason 4 from { 3 }
 5.  I     con-reason 4 from { 4 }
 7.  L     pf-reason 5 from { 5 }

================================================================================
ARGUMENT #2
This is a defeated argument for:
     K
 which is of ultimate interest.

 2.  B     given
 8.  E     pf-reason 3 from { 2 }
 12.  H     con-reason 2 from { 8 }
 13.  K     con-reason 3 from { 12 }

This argument is defeated by argument #5
================================================================================
ARGUMENT #3
This is an undefeated argument of strength 1 for:
     J
 which is of ultimate interest.

 1.  A     given
 9.  D     pf-reason 1 from { 1 }
 10.  G     pf-reason 2 from { 9 }
 11.  J     con-reason 1 from { 10 }

This argument is defeated by argument #4
================================================================================
ARGUMENT #4
This is a defeated argument for:
     (D @ G)

 2.  B     given
 8.  E     pf-reason 3 from { 2 }
 12.  H     con-reason 2 from { 8 }
 14.  (D @ G)     con-reason 6 from { 12 }

This argument is defeated by argument #5

This argument defeats argument #3
================================================================================
ARGUMENT #5
This is an undefeated argument of strength 1 for:
     (B @ E)

 3.  C     given
 4.  F     pf-reason 4 from { 3 }
 6.  (B @ E)     con-reason 5 from { 4 }
```

```
This argument defeats arguments #2, #4
============================================================================


Cumulative size of arguments = 14
Size of inference-graph = 14 of which 0 were unused suppositions.
100% of the inference-graph was used in the argument.
```

It will be illuminating to consider problem #2 in the problem set, which is a simple case of collective defeat:

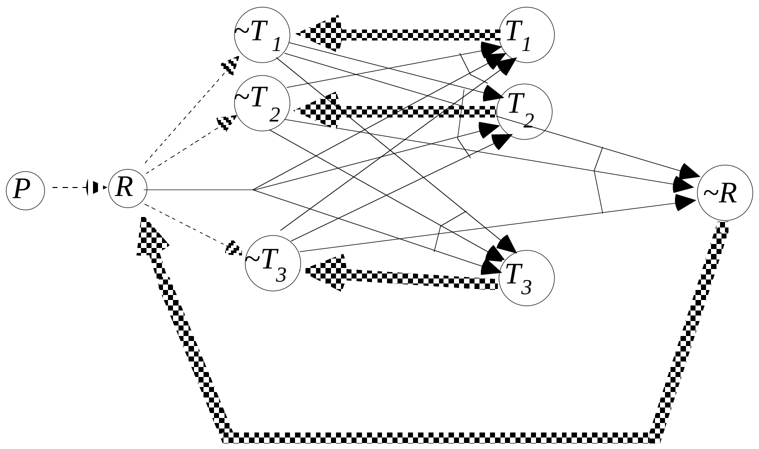


```
Problem #2
This is a case of collective defeat.
Given premises:
    P    justification = 1
    A    justification = 1
Ultimate epistemic interests:
    R    interest = 1

   FORWARDS PRIMA FACIE REASONS
     pf-reason 1:   {P} ||=> Q   strength = 1
     pf-reason 2:   {Q} ||=> R   strength = 1
     pf-reason 3:   {A} ||=> B   strength = 1

   BACKWARDS PRIMA FACIE REASONS
     pf-reason 4:   {} {C} ||=> ~R   strength = 1
     pf-reason 5:   {} {B} ||=> C   strength = 1

 # 1   P
 given
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1


 # 2   A
 given
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: R> from the inference-queue.
                              # 1
                              interest: R
                              This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
                              # 2
                              interest: (A @ B)
                              Of interest as defeater for support-link 1
                              # 3
                              interest: ~B
                              Of interest as defeater for support-link 1

 # 3   B
 Inferred by support-link #1 from { 2 } by pf-reason 3
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 1
 This inference is defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.
                              # 4
                              interest: (P @ Q)
                              Of interest as defeater for support-link 2
                              # 5
```

```
                                        interest: ~Q
                                        Of interest as defeater for support-link 2

  # 4   Q
  Inferred by support-link #2 from { 1 } by pf-reason 1
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
                                        # 6
                                        interest: (Q @ R)
                                        Of interest as defeater for support-link 3
                                        # 7
                                        interest: ~R
                                        Of interest as defeater for support-link 3

  # 5   R
  Inferred by support-link #3 from { 4 } by pf-reason 2
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
               ========================================
               Justified belief in R
               answers #<Query #1: R> affirmatively.
               ========================================
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 7: ~R supposing {  }> from the inference-queue.
                                        # 8
                                        interest: C
                                        For interest 7 by pf-reason 4
---------------------------------------------------------------------------
Retrieving #<Interest 8: C supposing {  }> from the inference-queue.
                                        # 9
                                        interest: B
                                        For interest 8 by pf-reason 5
                                        Conclusion #3 discharges interest #9
                                        # 10
                                        interest: (B @ C)
                                        Of interest as defeater for support-link 4
                                        # 11
                                        interest: ~C
                                        Of interest as defeater for support-link 4

  # 6   C
  Inferred by support-link #4 from { 3 } by pf-reason 5
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
  This node discharges interest 8
                                        # 12
                                        interest: (C @ ~R)
                                        Of interest as defeater for support-link 5
                                   Readopting interest in:
                                        # 1
                                        interest: R
                                        Of interest as defeater for support-link 5
  Node # 5 defeats link # 5
  Node # 7 defeats link # 3
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 6> . 1) (#<Node 4> . 1) (#<Node 3> . 1) (#<Node 2> . 1) (#<Node 1> . 1))
. triangle 1:
. . (#<support-link #5 for node 7> #<support-link #3 for node 5>)
. . . . assignment-tree 1:
. . . . . ((#<Node 5> . 0) (#<Node 7> . 1))
. . . . assignment-tree 2:
. . . . . ((#<Node 5> . 1) (#<Node 7> . 0))
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 5> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              ==========================================
              Lowering the undefeated-degrees-of-support of R
              retracts the previous answer to #<Query #1: R>
              ==========================================

  # 7   ~R                  DEFEATED
  Inferred by support-link #5 from { 6 } by pf-reason 4  defeaters: { 5 }
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is defeasible.
  defeatees: { 3 }
  This node discharges interest 7
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 9: B supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 11: ~C supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 5: ~Q supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 3: ~B supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 10: (B @ C) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 6: (Q @ R) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 4: (P @ Q) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 2: (A @ B) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 12: (C @ ~R) supposing {  }> from the inference-queue.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in R
  is unsatisfied.
---------------------------------------------------

Elapsed time = 3.93 sec

============================================================================
ARGUMENT #1
This is a defeated argument for:
      R
 which is of ultimate interest.

 1.  P     given
 4.  Q     pf-reason 1 from { 1 }
 5.  R     pf-reason 2 from { 4 }

This argument is defeated by argument #2

This argument defeats argument #2
============================================================================
ARGUMENT #2
This is a defeated argument for:
      ~R
 2.  A     given
 3.  B     pf-reason 3 from { 2 }
 6.  C     pf-reason 5 from { 3 }
 7.  ~R     pf-reason 4 from { 6 }

This argument is defeated by argument #1

This argument defeats argument #1
============================================================================


Cumulative size of arguments = 7
Size of inference-graph = 7 of which 0 were unused suppositions.
100% of the inference-graph was used in the argument.
```

Figure 4 of chapter one diagrammed the "Lottery Paradox Paradox". The following is a display of OSCAR's reasoning applied to that problem for the simplified case of three tickets:

```
Problem #8
Figure 8 -- the lottery paradox paradox
Given premises:
     P    justification = 1
Ultimate epistemic interests:
     ~T1    interest = 1
     ~T2    interest = 1
     ~T3    interest = 1

    FORWARDS PRIMA FACIE REASONS
      pf-reason 1:   {R} ||=> ~T1   strength = 1
      pf-reason 2:   {R} ||=> ~T2   strength = 1
      pf-reason 3:   {R} ||=> ~T3   strength = 1
      pf-reason 4:   {P} ||=> R   strength = 1

    FORWARDS CONCLUSIVE REASONS
      con-reason 1:   {R , ~T1 , ~T2} ||=> T3   strength = 1
      con-reason 2:   {R , ~T2 , ~T3} ||=> T1   strength = 1
      con-reason 3:   {R , ~T1 , ~T3} ||=> T2   strength = 1
      con-reason 4:   {~T1 , ~T2 , ~T3} ||=> ~R   strength = 1
```



```
  # 1   P
  given
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.
```

```
                                        # 1
                                        interest: (P @ R)
                                        Of interest as defeater for support-link 1
                                        # 2
                                        interest: ~R
                                        Of interest as defeater for support-link 1

  # 2   R
  Inferred by support-link #1 from { 1 } by pf-reason 4
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
                                        # 3
                                        interest: (R @ ~T1)
                                        Of interest as defeater for support-link 2
                                        # 4
                                        interest: T1
                                        Of interest as defeater for support-link 2

  # 3   ~T1
  Inferred by support-link #2 from { 2 } by pf-reason 1
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
                                        # 5
                                        interest: (R @ ~T2)
                                        Of interest as defeater for support-link 3
                                        # 6
                                        interest: T2
                                        Of interest as defeater for support-link 3

  # 4   ~T2
  Inferred by support-link #3 from { 2 } by pf-reason 2
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
                                        # 7
                                        interest: (R @ ~T3)
                                        Of interest as defeater for support-link 4
                                        # 8
                                        interest: T3
                                        Of interest as defeater for support-link 4

  # 5   ~T3
  Inferred by support-link #4 from { 2 } by pf-reason 3
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 1
  This inference is defeasible.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
  Node # 6 defeats link # 2
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 3> . 0) (#<Node 6> . 1) (#<Node 5> . 1) (#<Node 4> . 1) (#<Node 2> . 1)
.   (#<Node 1> . 1))
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 3> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

  # 6   T1
  Inferred by support-link #5 from { 4 , 2 , 5 } by con-reason 2
  maximal-degree-of-support: 1
```

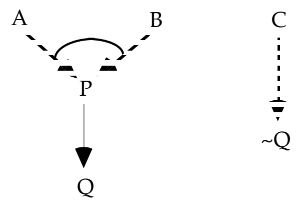
```
 undefeated-degree-of-support: 1
 This inference is not defeasible.
 defeatees: { 2 }
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Query #1: ~T1> from the inference-queue.
                                  # 9
                                  interest: ~T1
                                  This is of ultimate interest
             ----------------------------------------
             #<Node 3> answers #<Query #1: ~T1>
             ----------------------------------------
---------------------------------------------------------------------------
Retrieving #<Query #2: ~T2> from the inference-queue.
                                  # 10
                                  interest: ~T2
                                  This is of ultimate interest
             ----------------------------------------
             #<Node 4> answers #<Query #2: ~T2>
             ----------------------------------------
---------------------------------------------------------------------------
Retrieving #<Query #3: ~T3> from the inference-queue.
                                  # 11
                                  interest: ~T3
                                  This is of ultimate interest
             ----------------------------------------
             #<Node 5> answers #<Query #3: ~T3>
             ----------------------------------------
---------------------------------------------------------------------------
Retrieving #<Interest 8: T3 supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 6: T2 supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 4: T1 supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.
 Node # 7 defeats link # 4
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 4> . 1) (#<Node 2> . 1) (#<Node 1> . 1))
. triangle 1:
. . (#<support-link #4 for node 5> #<support-link #5 for node 6> #<support-link #2 for node 3>
#<support-link #6 for node 7>)
. . . . assignment-tree 1:
. . . . . ((#<Node 7> . 0) (#<Node 3> . 0) (#<Node 6> . 1) (#<Node 5> . 1))
. . . . assignment-tree 2:
. . . . . ((#<Node 7> . 1) (#<Node 3> . 1) (#<Node 6> . 0) (#<Node 5> . 0))
             vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
             #<Node 5> has become defeated.
             vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
             #<Node 6> has become defeated.
             vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
             ========================================
             Lowering the undefeated-degrees-of-support of ~T3
             retracts the previous answer to #<Query #3: ~T3>
             ========================================

 # 7   T3                  DEFEATED
 Inferred by support-link #6 from { 3 , 2 , 4 } by con-reason 1
 maximal-degree-of-support: 1
 undefeated-degree-of-support: 0
 This inference is not defeasible.
 defeatees: { 4 }
 Node # 8 defeats link # 3
---------------------------------------------------------------------------
```

```
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 2> . 1) (#<Node 1> . 1))
. triangle 2:
. . (#<support-link #4 for node 5> #<support-link #5 for node 6> #<support-link #7 for node 8>
#<support-link #3 for node 4> #<support-link #2 for node 3>
. .   #<support-link #6 for node 7>)
. . . . assignment-tree 3:
. . . . . ((#<Node 8> . 0) (#<Node 7> . 0) (#<Node 3> . 0) (#<Node 6> . 1) (#<Node 5> . 1)
. . . . .   (#<Node 4> . 1))
. . . . assignment-tree 4:
. . . . . ((#<Node 8> . 1) (#<Node 7> . 0) (#<Node 3> . 1) (#<Node 6> . 0) (#<Node 5> . 1)
. . . . .   (#<Node 4> . 0))
. . . . assignment-tree 5:
. . . . . ((#<Node 8> . 0) (#<Node 7> . 1) (#<Node 3> . 1) (#<Node 6> . 0) (#<Node 5> . 0)
. . . . .   (#<Node 4> . 1))
               vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
               #<Node 4> has become defeated.
               vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
               =========================================
               Lowering the undefeated-degrees-of-support of ~T2
               retracts the previous answer to #<Query #2: ~T2>
               =========================================

  # 8   T2                  DEFEATED
  Inferred by support-link #7 from { 3 , 2 , 5 } by con-reason 3
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is not defeasible.
  defeatees: { 3 }
  Node # 9 defeats link # 1
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 0:
. ((#<Node 1> . 1))
. triangle 3:
. . (#<support-link #1 for node 2> #<support-link #5 for node 6> #<support-link #7 for node 8>
#<support-link #6 for node 7> #<support-link #4 for node 5>
. .   #<support-link #3 for node 4> #<support-link #2 for node 3> #<support-link #8 for node 9>)
. . . . assignment-tree 6:
. . . . . ((#<Node 9> . 0) (#<Node 8> . 0) (#<Node 6> . 0) (#<Node 5> . 0) (#<Node 7> . 1)
. . . . .   (#<Node 2> . 1) (#<Node 3> . 1) (#<Node 4> . 1))
. . . . assignment-tree 7:
. . . . . ((#<Node 9> . 0) (#<Node 8> . 1) (#<Node 6> . 0) (#<Node 5> . 1) (#<Node 7> . 0)
. . . . .   (#<Node 2> . 1) (#<Node 3> . 1) (#<Node 4> . 0))
. . . . assignment-tree 8:
. . . . . ((#<Node 9> . 0) (#<Node 8> . 0) (#<Node 6> . 1) (#<Node 5> . 1) (#<Node 7> . 0)
. . . . .   (#<Node 2> . 1) (#<Node 3> . 0) (#<Node 4> . 1))

  # 9   ~R                  DEFEATED
  Inferred by support-link #8 from { 3 , 4 , 5 } by con-reason 4
  maximal-degree-of-support: 1
  undefeated-degree-of-support: 0
  This inference is not defeasible.
  defeatees: { 1 }
---------------------------------------------------------------------------
Retrieving #<Node 8> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 9> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 2: ~R supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 1: (P @ R) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 7: (R @ ~T3) supposing {  }> from the inference-queue.
```

```
---------------------------------------------------------------------------
Retrieving #<Interest 5: (R @ ~T2) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 3: (R @ ~T1) supposing {  }> from the inference-queue.


================= ULTIMATE EPISTEMIC INTERESTS ==================
  Interest in ~T3
  is unsatisfied.
---------------------------------------------------
  Interest in ~T2
  is unsatisfied.
---------------------------------------------------
  Interest in ~T1
  is unsatisfied.
---------------------------------------------------

Elapsed time = 6.05 sec

==============================================================================
ARGUMENT #1
This is a defeated argument for:
      ~T3
 which is of ultimate interest.

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 5.  ~T3     pf-reason 3 from { 2 }

This argument is defeated by arguments #6, #7
==============================================================================
ARGUMENT #2
This is a defeated argument for:
      ~T2
 which is of ultimate interest.

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 4.  ~T2     pf-reason 2 from { 2 }

This argument is defeated by arguments #5, #7
==============================================================================
ARGUMENT #3
This is a defeated argument for:
      ~T1
 which is of ultimate interest.

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 3.  ~T1     pf-reason 1 from { 2 }

This argument is defeated by arguments #4, #7
==============================================================================
ARGUMENT #4
This is a defeated argument for:
      T1

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 4.  ~T2     pf-reason 2 from { 2 }
 5.  ~T3     pf-reason 3 from { 2 }
 6.  T1     con-reason 2 from { 4 , 2 , 5 }

This argument is defeated by arguments #5, #6, #7

This argument defeats arguments #3, #5, #6, #7
==============================================================================
ARGUMENT #5
This is a defeated argument for:
     T2

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 3.  ~T1     pf-reason 1 from { 2 }
 5.  ~T3     pf-reason 3 from { 2 }
 8.  T2     con-reason 3 from { 3 , 2 , 5 }

This argument is defeated by arguments #4, #6, #7

This argument defeats arguments #2, #4, #6, #7
=============================================================================
ARGUMENT #6
This is a defeated argument for:
     T3

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 3.  ~T1     pf-reason 1 from { 2 }
 4.  ~T2     pf-reason 2 from { 2 }
 7.  T3     con-reason 1 from { 3 , 2 , 4 }

This argument is defeated by arguments #4, #5, #7

This argument defeats arguments #1, #4, #5, #7
=============================================================================
ARGUMENT #7
This is a defeated argument for:
     ~R

 1.  P     given
 2.  R     pf-reason 4 from { 1 }
 3.  ~T1     pf-reason 1 from { 2 }
 4.  ~T2     pf-reason 2 from { 2 }
 5.  ~T3     pf-reason 3 from { 2 }
 9.  ~R     con-reason 4 from { 3 , 4 , 5 }

This argument is defeated by arguments #4, #5, #6, #7

This argument defeats arguments #1, #2, #3, #4, #5, #6, #7
=============================================================================

Cumulative size of arguments = 9
Size of inference-graph = 9 of which 0 were unused suppositions.
100% of the inference-graph was used in the argument.
```

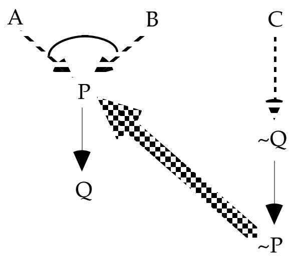
Although OSCAR performs well on the preceding problems, there are some simple problems on which OSCAR fails. Consider the following:

```
Problem #18
Given premises:
    B    justification = 1.0
    A    justification = 1.0
    C    justification = 1.0
Ultimate epistemic interests:
    Q    interest = 0.7

   FORWARDS PRIMA FACIE REASONS
     pf-reason 1:   {A , B} ||=> P   strength = 0.7
     pf-reason 2:   {C} ||=> ~Q   strength = 0.8

   FORWARDS CONCLUSIVE REASONS
     con-reason 1:   {P} ||=> Q   strength = 1.0
================== ULTIMATE EPISTEMIC INTERESTS ===================
```



```
 Interest in Q
 is satisfied.
===========================================================================
ARGUMENT #1
This is an undefeated argument of strength 0.7 for:
    Q
 which is of ultimate interest.

 1.  B     given
 2.  A     given
 5.  P     pf-reason 1 from { 1 , 2 }
 6.  Q     con-reason 1 from { 5 }

===========================================================================
```

Intuitively, *P* and *Q* should both be defeated in this example. However, the node supporting *~Q* cannot defeat the support-link for *Q*, because that proceeds via a nondefeasible reason. To get *P* and *Q* defeated, the reasoner must recognize that because *P* entails *Q*, *~Q* entails *~P*, and hence the inference-graph can be expanded to look as follows:



This will have the consequence that *P* is defeated by *~P*, and then *Q* is defeated because it is derived from *P*. On the other hand, *~Q* will not be defeated by *Q*, because the strength of the node supporting *Q* is only 0.7 whereas the strength of the node supporting *~Q* is 0.8.

The difficulty is that, as it stands, OSCAR does not automatically perform the additional reasoning required to generate the defeat in the above example. To rectify this, we must add some additional machinery to OSCAR. First, we keep track of c-lists whose formulas contradict each other by recording this in the new slot *c-list-contradictors* added to c-lists. When a new c-list is constructed, a search is made for contradicting c-lists and the contradiction is recorded in both contradicting c-lists. (Without unification, there will be at most one contradicting c-list, but when unification is incorporated in chapter five, it will become possible for there to be multiple contradicting c-lists.) This is done by STORE-INFERENCE-NODE. Contradictions are then handled by having REASON-FORWARDS-FROM apply INVERT-CONTRADICTIONS, which in the preceding example will result in OSCAR contraposing the entailment of *Q* from *P*, and infer *~P* from *~Q*.

Thus far I have only considered the case of contradictions between nodes having empty node-suppositions. Suppose instead that we have a node with the node-sequent *P*/*A*, and from that we infer *Q*/*B* deductively. It does not follow automatically that we can infer *~P*/*A* from *~Q*/*B*. Let Π*A* and Π*B* be the conjunctions of formulas in the suppositions *A* and *B* respectively. *P*/*A* is interderivable with (Π*A* ⊃ *P*), so if *B* ⊆ *A* then (Π*A* ⊃ Π*B*) holds and we can reason as follows:

```
 1.  ((ΠA ⊃ P) ⊃ (ΠB ⊃ Q))     given
 2.  (ΠA ⊃ ΠB)     given
     |----------------------------------------------------------
     | Suppose:  { ΠA }
     |----------------------------------------------------------
     | 4.  ΠA      supposition
     | 5.  ΠB     modus-ponens2 from { 4 , 2 }
```

```
    |------------------------------------------------------------
    | Suppose:  { (ΠB ⊃ ~Q) }
    |------------------------------------------------------------
    | 3.  (ΠB ⊃ ~Q)     supposition
          |------------------------------------------------------------
          | Suppose:  { ΠA , (ΠB ⊃ ~Q) }
          |------------------------------------------------------------
          | 6.  ~Q     modus-ponens2 from { 5 , 3 }
          | 8.  ~(ΠB ⊃ Q)     i-neg-condit from { 5 , 6 }
          | 9.  ~(ΠA ⊃ P)     modus-tollens2 from { 8 , 1 }
          | 10.  ~P     neg-condit from { 9 }
    | 11.  (ΠA ⊃ ~P)     conditionalization from { 10 }
12.  ((ΠB ⊃ ~Q) ⊃ (ΠA ⊃ ~P))     conditionalization from { 11 }
```

Thus we can infer $\sim P/A$ from $\sim Q/B$ provided $B \subseteq A$ (and not in general otherwise). More generally, given sequents $\sigma_1,...,\sigma_n$, if $\sigma_1,...,\sigma_n$, P/A entails Q/B and $B \subseteq A$ then $\sigma_1,...,\sigma_n$, $\sim Q/B$ entails $\sim P/A$. So suppose $B \subseteq A$ and we have an argument supporting Q/B with the nearest defeasible ancestors $\sigma_1,...,\sigma_n$, P/A, and another argument supporting $\sim Q/C$ with the nearest defeasible ancestors $\alpha_1,...,\alpha_m$. If $C \subseteq B$, then $\sim Q/C$ entails $\sim Q/B$, so $\alpha_1,...,\alpha_m,\sigma_1,...,\sigma_n$ entails $\sim P/A$. Accordingly, INVERT-CONTRADICTIONS becomes the following:

INVERT-CONTRADICTIONS *node*
- Let *c-list* be the c-list for*node*.
- If *nl* is a c-list-contradictor of *c-list*, then for every *node** in the c-list-nodes of *nl*:
  - If *node** is not a deductive-node and the node-supposition of *node** is contained in that of *node**, then for each nearest-defeasible-ancestor *D* of node, if the node-supposition of *node* is a subset of the node-supposition of *D* and *P/A* is the node-sequent of *D*, infer *~P/A* from *node** and any other nearest-defeasible-ancestors of *node*.
  - If *node* is not a deductive-node and the node-supposition of *node* is contained in that of *node**, then for each nearest-defeasible-ancestor *D* of *node**, if the node-supposition of *node** is a subset of the node-supposition of *D* and *P/A* is the node-sequent of *D*, infer *~P/A* from from *node* and any other nearest-defeasible-ancestors of *node**.

With this addition, OSCAR handles the preceding problem correctly as follows:

```
Problem #18
This uses contradiction-inversion.
Given premises:
   B    justification = 1.0
   A    justification = 1.0
   C    justification = 1.0
Ultimate epistemic interests:
   Q    interest = 0.7

  FORWARDS PRIMA FACIE REASONS
    pf-reason 1:   {A , B} ||=> P   strength = 0.7
    pf-reason 2:   {C} ||=> ~Q   strength = 0.8

  FORWARDS CONCLUSIVE REASONS
    con-reason 1:   {P} ||=> Q   strength = 1.0

================= ULTIMATE EPISTEMIC INTERESTS ==================
 Interest in Q
 is unsatisfied.
=========================================================================
ARGUMENT #1
This is a defeated argument for:
    Q
 which is of ultimate interest.

 1.  B     given
 2.  A     given
 5.  P     pf-reason 1 from { 1 , 2 }
 6.  Q     con-reason 1 from { 5 }

Argument #2 support defeaters for this argument.

This argument supports defeaters for argument #2
=============================================================================
ARGUMENT #2
This is an undefeated argument of strength 0.8 for:
   ~P

 6.  Q     by argument #1
 3.  C     given
 4.  ~Q     pf-reason 2 from { 3 }
 7.  ~P    inversion from contradictory nodes 6 and 4 from { 4 }

Argument #1 support defeaters for this argument.

This argument supports defeaters for argument #1
=============================================================================
```

2. The Propositional Calculus

OSCAR can be made to prove theorems in the propositional calculus by supplying appropriate reason schemas.

It turns out to be convenient to formulate inference rules (conclusive reason schemas for logic) using '¬', where we define ⌜¬ p⌝ to be q if p is ⌜~q⌝, and to be ⌜~p⌝ otherwise. Then the following set of inference rules (conclusive reasons) is complete for the propositional calculus:

Forwards reasons:

simplification: infer p and q from (p & q)
negation-elimination: infer p from ~~p
disjunction-negation: infer ¬p and ¬q from ~($p \vee q$)
conditional-negation: infer p and ¬q from ~($p \to q$)
DeMorgan: infer (¬$p \vee$ ¬q) from ~(p & q)
biconditional-simplification: infer ($p \to q$) and ($q \to p$) from ($p \leftrightarrow q$)
biconditional-negation-simplication : infer ($p \leftrightarrow$ ¬q) from ~($p \leftrightarrow q$)
modus-ponens: infer q from p and ($p \to q$)
modus-tollens: infer ¬p from ($p \to q$) and ¬q
disjunctive syllogism: infer q from ($p \vee q$) and ¬p
infer p from ($p \vee q$) and ¬q
exportation : infer ($p \to (q \to r)$) from ((p & q) $\to r$)
disjunction-simplification: infer ($p^* \to q$) from ($p \vee q$), where p^* is the result of negating p and driving the negation in across all truth-functional connectives.

Backwards reasons:

adjunction: adopt interest in p and q to infer (p & q)
negation introduction: adopt interest in p to infer ~~p
disjunction negation introduction:
adopt interest in ¬p and¬q to infer ~($p \vee q$)
conditional negation introduction:
adopt interest in p and¬q to infer ~($p \to q$)
biconditional introduction:
adopt interest in ($p \to q$) and in ($q \to p$) to infer ($p \leftrightarrow q$)
biconditional negation introduction:
adopt interest in ($p \leftrightarrow$ ¬q) to infer ~($p \leftrightarrow q$)

disjunction introduction: adopt interest in $(\neg p \rightarrow q)$ to infer $(p \vee q)$
backwards DeMorgan: adopt interest in $(\neg p \vee \neg q)$ to infer $\sim(p \ \& \ q)$
conditionalization: adopt interest in $q/X\cup\{p\}$ to infer $(p \rightarrow q)/X$
reductio: suppose (i.e., draw the conclusion) $\neg p/\{\neg p\}$, and then for any q, if $q/X\cup\{\neg p\}$ is concluded, adopt interest in $\neg q\rangle/X\cup\{\neg p\}$ with a discount-factor **reductio-discount**, to infer p/X, provided p is a literal (an atomic formula or the negation of an atomic formula), disjunction, or conditional, and $\neg p$ is not deductively validated in X, and q is either a literal or a conditional.

These rules are easily encoded using the macros def-fowards-reason and def-backwards-reason. For example, *simplification* can be defined as follows:

```
(def-forwards-reason simplification
    :forwards-premises  "(P & Q)"
    :conclusions   "P" "Q"
    :variables  P Q)
```

Similarly, *adjunction* can be defined as follows:

```
(def-backwards-reason adjunction
    :conclusion  "(P & Q)"
    :forwards-premises  "P" "Q"
    :variables  P Q)
```

Conditionalization is defined as follows:

```
(def-backwards-reason conditionalization
    :conclusion "(P -> Q)"
    :forwards-premises  "Q"
    :discharge  "P"
    :variables  P Q)
```

The most complex of these reasons is *reductio*, which is defined as follows:

```
(def-backwards-reason reductio
    :conclusion "P"
    :condition  (or (literalp P) (disjunctionp P) (conditionalp P))
    :forwards-premises
        "Q"
        (:condition  (or (literalp Q) (conditionalp Q)))
    :backwards-premises
         "~Q"
    :discharge  "~P"
    :variables  P Q
    :discount-factor *reductio-discount*)
```

Notice that, as discussed in chapter three, *reductio* is a generalized backwards reason. Syntactical restrictions are imposed upon both p and q in *reductio* to prevent their duplicating reasoning performed in accordance with the other rules of inference. In addition, *reductio* has a discount-factor **reductio-discount** built into it. The value 0.25 has been chosen for **reductio-discount** by experiment. The use of such a discount-factor has the effect of making *reductio* a "last resort" strategy. Reasoning by *reductio* is only attempted when other reasoning strategies have failed.

OSCAR's performance on deductive reasoning in the propositional calculus is illustrated by problems #21 - 37 in *Problems.lsp*. The following is a typical example:

```
Problem #30

Given premises:
     (q -> r)    justification = 1.0
     (r -> (p & q))    justification = 1.0
     (p -> (q v r))    justification = 1.0
Ultimate epistemic interests:
     (p <-> q)    interest = 1.0


  # 1   (q -> r)
  given

  # 2   (r -> (p & q))
  given

  # 3   (p -> (q v r))
  given
                                        # 1
                                        interest: (p <-> q)
                                        This is of ultimate interest
                                        # 2
                                        interest: (q -> p)
                                        For interest 1 by bicondit-intro
---------------------------------------------------------------------------
1:    Retrieving #<Interest 2: (q -> p)> from the inference-queue.  Preference = 0.3333

  # 4   q    supposition: { q }
  supposition
                                        # 3
                                        interest: p    supposition: { q }
                                        For interest 2 by conditionalization
---------------------------------------------------------------------------
2:    Retrieving #<Interest 3: p supposing { q }> from the inference-queue.  Preference = 1.0

  # 5   ~p    supposition: { ~p }
  supposition
---------------------------------------------------------------------------
3:    Retrieving #<Node 4> from the inference-queue.  Preference = 1.0
                                        # 4
                                        interest: ~q    supposition: { ~p , q }
                                        For interest 3 by reductio using node 4
---------------------------------------------------------------------------
4:    Retrieving #<Node 1> from the inference-queue.  Preference = 0.3333
                                        # 5
                                        interest: ~(q -> r)    supposition: { ~p , q }
                                        For interest 3 by reductio using node 1
                                        # 6
                                        interest: ~r    supposition: { ~p , q }
                                        For interest 3 by reductio using node 6

  # 6   r    supposition: { q }
  Inferred by support-link #4 from { 1 , 4 } by modus-ponens1
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
5:    Retrieving #<Node 6> from the inference-queue.  Preference = 1.0
                                    Readopting interest in:
                                        # 6
                                        interest: ~r    supposition: { ~p , q }
                                        For interest 3 by reductio using node 6
                                        For interest 3 by reductio using node 6
---------------------------------------------------------------------------
6:    Retrieving #<Node 5> from the inference-queue.  Preference = 0.25
                                        # 7
                                        interest: p    supposition: { ~p , q }
                                        For interest 3 by reductio using node 5
---------------------------------------------------------------------------
```

```
7:    Retrieving #<Interest 7: p supposing { ~p , q }> from the inference-queue.  Preference = 0.25
---------------------------------------------------------------------------
8:    Retrieving #<Node 3> from the inference-queue.  Preference = 0.2
                                      # 8
                                      interest: ~(p -> (q v r))    supposition: { ~p , q }
                                      For interest 3 by reductio using node 3
---------------------------------------------------------------------------
9:    Retrieving #<Node 2> from the inference-queue.  Preference = 0.2
                                      # 9
                                      interest: ~(r -> (p & q))    supposition: { ~p , q }
                                      For interest 3 by reductio using node 2

  # 7   (p & q)    supposition: { q }
  Inferred by support-link #5 from { 2 , 6 } by modus-ponens1
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
10:    Retrieving #<Node 7> from the inference-queue.  Preference = 0.3333
  Node 8 discharges interest 7
..............................................................................................................................
Cancelling interest 7
. Cancelling  #<Interest 7: p supposing { ~p , q }>
..............................................................................................................................
  Node 8 discharges interest 3
  Node 9 discharges interest 2
                                      # 10
                                      interest: (p -> q)
                                      For interest 1 by bicondit-intro using node 9
..............................................................................................................................
Cancelling interest 2
. Cancelling  #<Node 5>
. Cancelling  #<Node 8>
. Cancelling  #<Node 7>
. Cancelling  #<Node 6>
. Cancelling  #<Node 4>
. Cancelling  #<Interest 4: ~q supposing { ~p , q }>
. Cancelling  #<Interest 5: ~(q -> r) supposing { ~p , q }>
. Cancelling  #<Interest 6: ~r supposing { ~p , q }>
. Cancelling  #<Interest 8: ~(p -> (q v r)) supposing { ~p , q }>
. Cancelling  #<Interest 9: ~(r -> (p & q)) supposing { ~p , q }>
. Cancelling  #<Interest 3: p supposing { q }>
. Cancelling  #<Interest 2: (q -> p)>
..............................................................................................................................

  # 8   p    supposition: { q }
  Inferred by support-link #6 from { 7 } by simp
  undefeated-degree-of-support = 1.0

  # 9   (q -> p)
  Inferred by support-link #7 from { 8 } by conditionalization
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
11:    Retrieving #<Node 8> from the inference-queue.  Preference = 0.5
---------------------------------------------------------------------------
12:    Retrieving #<Node 9> from the inference-queue.  Preference = 0.3333
---------------------------------------------------------------------------
13:    Retrieving #<Interest 10: (p -> q)> from the inference-queue.  Preference = 0.3333

  # 10   p    supposition: { p }
  supposition
                                      # 11
                                      interest: q    supposition: { p }
                                      For interest 10 by conditionalization
---------------------------------------------------------------------------
14:    Retrieving #<Interest 11: q supposing { p }> from the inference-queue.  Preference = 1.0

  # 11   ~q    supposition: { ~q }
  supposition
```

```
                                        # 12
                                        interest: ~(p -> (q v r))    supposition: { ~q , p }
                                        For interest 11 by reductio using node 3
                                        # 13
                                        interest: ~(r -> (p & q))    supposition: { ~q , p }
                                        For interest 11 by reductio using node 2
                                        # 14
                                        interest: ~(q -> p)    supposition: { ~q , p }
                                        For interest 11 by reductio using node 9
                                        # 15
                                        interest: ~(q -> r)    supposition: { ~q , p }
                                        For interest 11 by reductio using node 1
---------------------------------------------------------------------------
15:    Retrieving #<Node 10> from the inference-queue.  Preference = 1.0
                                        # 16
                                        interest: ~p    supposition: { ~q , p }
                                        For interest 11 by reductio using node 10

  # 12   (q v r)    supposition: { p }
  Inferred by support-link #8 from { 3 , 10 } by modus-ponens2
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
16:    Retrieving #<Node 12> from the inference-queue.  Preference = 0.3333
                                        # 17
                                        interest: ~(~q -> r)    supposition: { ~q , p }
                                        For interest 11 by reductio using node 13

  # 13   (~q -> r)    supposition: { p }
  Inferred by support-link #9 from { 12 } by disj-simp
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
17:    Retrieving #<Node 13> from the inference-queue.  Preference = 0.25
                                    Readopting interest in:
                                        # 17
                                        interest: ~(~q -> r)    supposition: { ~q , p }
                                        For interest 11 by reductio using node 13
                                        For interest 11 by reductio using node 13
---------------------------------------------------------------------------
18:    Retrieving #<Node 11> from the inference-queue.  Preference = 0.25
                                        # 18
                                        interest: q    supposition: { ~q , p }
                                        For interest 11 by reductio using node 11
                                        # 19
                                        interest: ~r    supposition: { ~q , p }
                                        For interest 11 by reductio using node 14

  # 14   r    supposition: { p , ~q }
  Inferred by support-link #10 from { 13 , 11 } by modus-ponens2
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
19:    Retrieving #<Node 14> from the inference-queue.  Preference = 1.0
                                    Readopting interest in:
                                        # 19
                                        interest: ~r    supposition: { ~q , p }
                                        For interest 11 by reductio using node 14
                                        For interest 11 by reductio using node 14

  # 15   (p & q)    supposition: { p , ~q }
  Inferred by support-link #11 from { 2 , 14 } by modus-ponens2
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
20:    Retrieving #<Node 15> from the inference-queue.  Preference = 0.3333
  Node 16 discharges interest 18
  Node 17 discharges interest 18
.............................................................................................................................
Cancelling interest 18
. Cancelling  #<Interest 18: q supposing { ~q , p }>
```

```
........................................................................................................................................
  Node 17 discharges interest 11
  Node 18 discharges interest 10
  Node 19 discharges interest 1

  # 16   q    supposition: { p , ~q }
  Inferred by support-link #12 from { 15 } by simp
  undefeated-degree-of-support = 1.0

  # 17   q    supposition: { p }
  Inferred by support-link #13 from { 11 , 16 } by reductio
  undefeated-degree-of-support = 1.0

  # 18   (p -> q)
  Inferred by support-link #14 from { 17 } by conditionalization
  undefeated-degree-of-support = 1.0

  # 19   (p <-> q)
  Inferred by support-link #15 from { 9 , 18 } by bicondit-intro
  undefeated-degree-of-support = 1.0

             ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (p <-> q)
  is answered affirmatively by node 19
---------------------------------------------------

==============================================================================
ARGUMENT #1
This is an undefeated argument of strength 1.0 for:
      (p <-> q)
 which is of ultimate interest.

 2.  (r -> (p & q))     given
 3.  (p -> (q v r))     given
 1.  (q -> r)     given
        |---------------------------------------------------------
        | Suppose:  { q }
        |---------------------------------------------------------
        | 4.  q     supposition
        | 6.  r     modus-ponens1 from { 1 , 4 }
        | 7.  (p & q)     modus-ponens1 from { 2 , 6 }
        | 8.  p     simp from { 7 }
 9.  (q -> p)     conditionalization from { 8 }
        |---------------------------------------------------------
        | Suppose:  { p }
        |---------------------------------------------------------
        | 10.  p     supposition
        | 12.  (q v r)     modus-ponens2 from { 3 , 10 }
        | 13.  (~q -> r)     disj-simp from { 12 }
        |---------------------------------------------------------
        | Suppose:  { ~q }
        |---------------------------------------------------------
        | 11.  ~q     supposition
                |---------------------------------------------------------
                | Suppose:  { p , ~q }
                |---------------------------------------------------------
                | 14.  r     modus-ponens2 from { 13 , 11 }
                | 15.  (p & q)     modus-ponens2 from { 2 , 14 }
                | 16.  q     simp from { 15 }
        |---------------------------------------------------------
        | Suppose:  { p }
        |---------------------------------------------------------
        | 17.  q     reductio from { 11 , 16 }
 18.  (p -> q)     conditionalization from { 17 }
 19.  (p <-> q)     bicondit-intro from { 9 , 18 }

============================================================================


Cumulative size of arguments = 18
Size of inference-graph = 19
94% of the inference-graph was used in the argument.
19 interests were adopted.
4 suppositions were made.
```

3. Parallelizing Reductio

The reason-schema *reductio* was defined very simply using DEF-BACKWARDS-REASON, but it turns out that the reason-schema thus produced exhibits some inherent inefficiencies. This is illustrated by the following problem:

```
Problem #29

Given premises:
Ultimate epistemic interests:
     (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))    interest = 1.0

                                        # 1
                                        interest: (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
                                        This is of ultimate interest

  # 1   ((p v q) & ((~p v q) & (p v ~q)))    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  supposition
                                        # 2
                                        interest: ~(~p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 1 by conditionalization
---------------------------------------------------------------------------
1:    Retrieving #<Node 1> from the inference-queue.  Preference = 1.0

  # 2   (p v q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #1 from { 1 } by simp
  undefeated-degree-of-support = 1.0

  # 3   ((~p v q) & (p v ~q))    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #2 from { 1 } by simp
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
2:    Retrieving #<Node 2> from the inference-queue.  Preference = 0.3333

  # 4   (~p -> q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #3 from { 2 } by disj-simp
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
3:    Retrieving #<Node 4> from the inference-queue.  Preference = 0.25
---------------------------------------------------------------------------
4:    Retrieving #<Interest 2: ~(~p v ~q) supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the
inference-queue.  Preference = 0.1666
                                        # 3
                                        interest: p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 2 by i-neg-disj
---------------------------------------------------------------------------
5:    Retrieving #<Interest 3: p supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-queue.
Preference = 1.0

  # 5   ~p    supposition: { ~p }
  supposition
                                        # 4
                                        interest: ~(~p -> q)    supposition: { ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 3 by reductio using node 4
------------------------------------------------------------------------
6:    Retrieving #<Node 5> from the inference-queue.  Preference = 0.25
                                        # 5
                                        interest: p    supposition: { ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 3 by reductio using node 5
                                        # 6
                                        interest: ~q    supposition: { ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 3 by reductio using node 6

  # 6   q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~p }
  Inferred by support-link #4 from { 4 , 5 } by modus-ponens2
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
7:    Retrieving #<Node 6> from the inference-queue.  Preference = 1.0
------------------------------------------------------------------------
8:    Retrieving #<Interest 5: p supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-
queue.  Preference = 0.25
------------------------------------------------------------------------
9:    Retrieving #<Interest 6: ~q supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-
queue.  Preference = 0.125

  # 7   q    supposition: { q }
  supposition
                                        # 7
                                        interest: ~q    supposition: { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 6 by reductio using node 6
                                        # 8
                                        interest: p    supposition: { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 6 by reductio using node 5
                                        # 9
                                        interest: ~(~p -> q)    supposition: { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 6 by reductio using node 4
------------------------------------------------------------------------
10:    Retrieving #<Node 7> from the inference-queue.  Preference = 0.25
------------------------------------------------------------------------
11:    Retrieving #<Node 3> from the inference-queue.  Preference = 0.1111

  # 8   (~p v q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #5 from { 3 } by simp
  undefeated-degree-of-support = 1.0

  # 9   (p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #6 from { 3 } by simp
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
12:    Retrieving #<Node 8> from the inference-queue.  Preference = 0.25
                                        # 10
                                        interest: ~(p -> q)    supposition: { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 6 by reductio using node 10
                                        # 11
                                        interest: ~(p -> q)    supposition: { ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 3 by reductio using node 10

  # 10   (p -> q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #7 from { 8 } by disj-simp
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
13:    Retrieving #<Node 10> from the inference-queue.  Preference = 0.3333
------------------------------------------------------------------------
14:    Retrieving #<Node 9> from the inference-queue.  Preference = 0.25
                                        # 12
                                        interest: ~(~p -> ~q)    supposition: { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 6 by reductio using node 11
                                        # 13
                                        interest: ~(~p -> ~q)    supposition: { ~p , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 3 by reductio using node 11
```

```
 # 11   (~p -> ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
 Inferred by support-link #8 from { 9 } by disj-simp
 undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
15:    Retrieving #<Node 11> from the inference-queue.  Preference = 0.2
 Node 12 discharges interest 7
.................................................................................................................................
Cancelling interest 7
. Cancelling  #<Interest 7: ~q supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
.................................................................................................................................
 Node 12 discharges interest 6
 Node 13 discharges interest 8
.................................................................................................................................
Cancelling interest 8
. Cancelling  #<Interest 8: p supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
.................................................................................................................................
 Node 13 discharges interest 5
.................................................................................................................................
Cancelling interest 5
. Cancelling  #<Interest 5: p supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
.................................................................................................................................
 Node 13 discharges interest 3
                                  # 14
                                  interest: q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                  For interest 2 by i-neg-disj using node 13
.................................................................................................................................
Cancelling interest 3
. Cancelling  #<Node 7>
. Cancelling  #<Node 6>
. Cancelling  #<Node 12>
. Cancelling  #<Node 5>
. Cancelling  #<Interest 4: ~(~p -> q) supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 9: ~(~p -> q) supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 10: ~(p -> q) supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 12: ~(~p -> ~q) supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 6: ~q supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 11: ~(p -> q) supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 13: ~(~p -> ~q) supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 3: p supposing { ((p v q) & ((~p v q) & (p v ~q))) }>
.................................................................................................................................

 # 12   ~q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~p }
 Inferred by support-link #9 from { 11 , 5 } by modus-ponens1
 undefeated-degree-of-support = 1.0

 # 13   p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
 Inferred by support-link #10 from { 6 , 12 } by reductio
 undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
16:    Retrieving #<Interest 14: q supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-
queue.  Preference = 1.0

 # 14   ~q    supposition: { ~q }
 supposition
                                  # 15
                                  interest: ~(p -> q)    supposition: { ~q , ((p v q) & ((~p v q) & (p v ~q))) }
                                  For interest 14 by reductio using node 10
                                  # 16
                                  interest: ~(~p -> ~q)    supposition: { ~q , ((p v q) & ((~p v q) & (p v ~q))) }
                                  For interest 14 by reductio using node 11
                                  # 17
                                  interest: ~(~p -> q)    supposition: { ~q , ((p v q) & ((~p v q) & (p v ~q))) }
                                  For interest 14 by reductio using node 4
---------------------------------------------------------------------------
17:    Retrieving #<Node 14> from the inference-queue.  Preference = 0.25
                                  # 18
```

```
                                        interest: q    supposition: { ~q , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 14 by reductio using node 14
                                        # 19
                                        interest: p    supposition: { ~q , ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 14 by reductio using node 15
                                        Node #13 discharges interest #19
  Node 16 discharges interest 18
.................................................................................................................................
Cancelling interest 18
. Cancelling  #<Interest 18: q supposing { ~q , ((p v q) & ((~p v q) & (p v ~q))) }>
.................................................................................................................................
  Node 16 discharges interest 14
  Node 17 discharges interest 2
  Node 18 discharges interest 1

  # 15   ~p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~q }
  Inferred by support-link #11 from { 10 , 14 } by modus-tollens2
  undefeated-degree-of-support = 1.0

  # 16   q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #12 from { 15 , 13 } by reductio
  undefeated-degree-of-support = 1.0

  # 17   ~(~p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #13 from { 13 , 16 } by i-neg-disj
  undefeated-degree-of-support = 1.0

  # 18   (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
  Inferred by support-link #14 from { 17 } by conditionalization
  undefeated-degree-of-support = 1.0

            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
  is answered affirmatively by node 18
---------------------------------------------------

Cumulative size of arguments = 17
Size of inference-graph = 18 of which 0 were unused suppositions.
94% of the inference-graph was used in the argument.
19 interests were adopted.
4 suppositions were made.
```

Observe that *reductio* generates four interests in p relative to different suppositions, 5 interests in ~(~p -> q) relative to different suppositions, two interests in ~q relative to different suppositions, and three interests in ~(p -> q) relative to different suppositions. What is happening here is that the conclusions p, (~p -> q), q, and (p -> q) have been drawn, so each time a new reductio supposition is made, interest is adopted in their negations relative to that supposition. This then produces essentially the same backwards reasoning from each interest with the same interest-formula. For instance, interest in ~(p -> q) relative to different suppositions will produce interests in p and ~q relative to each of those suppositions.

It seems that this backwards reasoning from "analogous" interests could all be done in parallel. Or more accurately, it could be done just once using a "variable-reductio-supposition", and then the resulting interests can be discharged multiple times by instantiating the variable-reductio-supposition differently for different discharging conclusions. This can be implemented by introducing a slot in interests for "reductio-interest". A reductio-interest will be any interest introduced by *reductio* or derived by reasoning backwards from another reductio-interest. We can modify the rule of *reductio* so that the reductio-supposition is no longer included in the interest-supposition of a reductio-interest. Thus a single reductio-interest can encode interest in the same sequent relative to many different reductio-suppositions.

However, this is not yet a complete solution to the problem. This will minimize the adoption of reductio-interests, but the same reductio-interests will be adopted repeatedly with different interest-links in attempts to get different non-reductio-interests. If the reasoner concludes *P* and then adopts interest in ~*P* in order to get some other interest *Q* (recording that in one interest-link), it will also adopt interest in ~*P* to get any other appropriate interest *R* (recording that in another interest-link). Thus we have tamed the profusion of interests, but we still get a profusion of interest-links. What this illustrates is that the use of interest-links is unnecessary in reductio-reasoning, because that reasoning generalizes across a wide range of interests. This suggests treating such reasoning differently than we treat reasoning in accordance with other interest rules. We can think of such reasoning as consisting of three coordinated operations. First, we have an operation driven by drawing new conclusions. Whenever we conclude a sequent of an appropriate syntactic type, we construct a direct-reductio-interest in its negation. This will be done by ADOPT-REDUCTIO-INTEREST. Second, given an interest of an appropriate syntactic type, we make a reductio-supposition of its negation. This is done by MAKE-REDUCTIO-SUPPOSITION. Third, when (1) a new inference-node has a reductio-supposition among its inference-ancestors, (2) its node-formula is that of a reductio-interest, and (3) its non-reductio-supposition is a subset of the supposition of the reductio-interest, then we can infer the negation of the reductio-supposition. This will be done by DISCHARGE-REDUCTIOS and DISCHARGE-RETROSPECTIVE-REDUCTIOS.

To implement schematic reductio-reasoning, a *reductio-supposition* will be an inference-node whose node-rule is "reductio-supposition". Such nodes record the making of suppositions for reductio-reasoning. It will be convenient to keep a record of the reductio-suppositions that are ancestors of a node in the new slot "reductio-ancestors". A reductio-supposition is its own only reductio-ancestor. The reductio-ancestors of any other node produced by a rule other than DISCHARGE-REDUCTIOS will be the reductio-ancestors of the members of the node-basis. DISCHARGE-REDUCTIOS and DISCHARGE-RETROSPECTIVE-REDUCTIOS will remove one node from the list of reductio-ancestors.

To parallelize reductio-reasoning, we must distinguish between the part of a supposition that is introduced by reductio-reasoning and the part that comes from other sources (e.g., conditionalization). The *non-reductio-supposition* of an inference node is the difference between the node-supposition and the set of negations of node-formulas of its reductio-ancestors. Rather than recomputing this repeatedly, we have a slot in inference-nodes for this. All told, we will add three slots to inference-nodes:

- reductio-ancestors — this is a list of pairs $\langle P,M\rangle$ where *M* is an inference-node and *P* is the node-formula of *M* or an instantiation of the node-formula.
- non-reductio-supposition — this is a list of pairs $\langle P,M\rangle$ where *M* is an inference-node and *P* is the node-formula of *M* or an instantiation of the node-formula. In the absence of reductio-reasoning, the node-supposition is the set of these *P*'s. In the presence of reductio-reasoning, the node-supposition is the set of *P*'s from the non-reductio-supposition and the list of reductio-ancestors, combined. These two slots provide a mechanism for tracing the souce of the node-supposition;
- reductios-discharged — T if the conclusion has already been used to discharge reductio-interests, and NIL otherwise;

Interests will still have slots for "reductio-interest" and "direct-reductio-interest". The latter will no longer have values of T or NIL, but will instead be either NIL or contain the list of generating conclusions whose negations are the objects of interest. There can be more than one such conclusion, because the supposition of a reductio-interest generated by a conclusion will be only the non-reductio-supposition of the conclusion:

- reductio-interest — T if the interest derives exclusively from attempts to use *reductio ad absurdum*; NIL otherwise.
- direct-reductio-interest — a list of inference-nodes.

It was indicated above that there are syntactic restrictions on the making of reductio-suppositions and the adoption of reductio-interests. This is to avoid duplication of effort. For example, OSCAR should never try to establish a conditional by reductio. Suppose OSCAR is interested in (P -> Q). The reasoner will suppose P and adopt interest in Q/{P}.

If OSCAR also tried to get (P -> Q) by reductio, it would suppose ~(P -> Q) and try to infer a contradiction. That is equivalent to supposing P and supposing ~Q and trying to get a contradiction, and that in turn is equivalent to supposing P for conditionalization and then trying to get Q by reductio. Thus to avoid duplication of effort, OSCAR should postpone the use of reductio until it has already supposed the antecedent of the conditional. For similar reasons, OSCAR should not make reductio-suppositions that are negations of biconditionals, conjunctions, disjunctions, or negations of truth-functional compounds. Similarly, OSCAR should not adopt reductio-interest in conjunctions, biconditionals, or negations of biconditionals, conjunctions, disjunctions, or negations of truth-functional compounds. If the correct inference-rules are chosen, so that forwards reasoning from any of these forms and their negation will lead to another contradiction involving subformulas, then the same restriction can be imposed on constructing the c-list-contradictors of a c-list. This will be particularly useful, because the c-list-contradictors are precisely the corresponding c-lists for the i-lists of reductio-interests. This allows us to avoid searching the entire list *interests* when we adopt a reductio-interest. We can instead just check the i-lists generated in this way.

The reasoner can check whether a formula is of the right form to generate a reductio-interest when its c-list is first constructed. Then it doesn't have to do it again. This information will be recorded in the slot "reductio-interests" in the c-list. Similarly, the reasoner will check whether a formula is of the right form to generate a reductio-supposition when its i-list is first constructed, and record this information in the slot "reductio-trigger" in the i-list. When the interests of an i-list lead to the making of a reductio-supposition, it is recorded in the new slot "i-list-reductio-supposition", and then when new interests are added to the i-list, they are also made generating-interests of the reductio-supposition.

When an interest is retrieved from the inference-queue, the reasoner checks to see the value of "reductio-trigger" for its i-list. If it is T, a reductio-supposition is made for the negation of the interest-formula, and "reductio-trigger" is reset to NIL. This must also be done when reasoning backwards from a query, because the resulting interest is not queued. A complication arises from the fact that a supposition can be made twice, both for reductio- and non-reductio-purposes (e.g., conditionalization). To accommodate that, we could create duplicate supposition-nodes, but that would create duplicate reasoning. Instead, OSCAR cancels the deductive-only status of reductio-suppositions when they are readopted as non-reductio-suppositions. In that case, the reductio-ancestors of the basis of an inference are only inherited by the conclusion as reductio-ancestors if the inference is deductive. Otherwise they are inherited as non-reductio-suppositions. When a non-reductio-supposition is readopted as a reductio-supposition, for all of its inclusive-node-descendants that are deductive in it, it is moved from the non-reductio-supposition to the list of reductio-ancestors. For all of those altered nodes that are not still on the inference-queue, we discharge-interest-in them and reason-forwards-from them. When a reductio-supposition is readopted as a non-reductio-supposition, it and all of its inclusive-node-descendants should cease to be deductive-only except insofar as they are inferred from other deductive-only nodes. We must also apply all members of defeasible forwards-reasons to it and its node-consequences.

MAKE-REDUCTIO-SUPPOSITION *interest*.
If the value of the reductio-trigger for the interest-i-list of *interest* is T, then:
- Reset the reductio-trigger to NIL.
- Where the interest-formula of *interest* is *q*, if there is an existing supposition *sup* supposing *q*, then if it is a non-reductio-supposition, convert it to a reductio-supposition. If there is no such supposition:
  - If *interest* is a non-reductio-interest, then for every non-reductio-supposition-node generating part of the interest-supposition, add that node to the list of *inherited-non-reductio-suppositions*.
  - Construct an inference-node *node* whose node-sequent is $\neg q/\{\neg q\}$ and insert it into the *inference-queue*. Let *node* be its own only reductio-ancestor.
  - Make *node* the i-list-reductio-supposition of the i-list of *interest*.
  - START-REDUCTIO-INTERESTS *N*.

Direct-reductio-interests are adopted under three different conditions. When reductio-suppositions are made, the nodes encoding them are stored in the list *reductio-supposition-nodes*. When the first reductio-supposition is made, OSCAR surveys all processed-conclusions, and adopts direct-reductio-interests in the negations of all those that are syntactically appropriate. In addition, whenever OSCAR makes a new reductio-supposition, OSCAR adopts a direct reductio-interest in its negation provided that it is syntactically appropriate. Finally, if the list *reductio-supposition-nodes* is nonempty, then when OSCAR retrieves a syntactically appropriate conclusion from the *inference-queue*, OSCAR adopts a direct-reductio-interest in its negation. Every reductio-supposition is a generating-node for a direct-reductio-interest. Accordingly, the priority of the new interest should be a function of the maximum discounted-node-strength for any reductio-supposition. This will be discounted by a constant factor **reductio-interest**, which experiment suggests setting to 0.25.

The above is accomplished by having MAKE-REDUCTIO-SUPPOSITION call START-REDUCTIO-INTERESTS, and having REASON-FORWARDS-FROM call ADOPT-REDUCTIO-INTEREST:

START-REDUCTIO-INTERESTS *node*.

- Add *node* to the list of *reductio-supposition-nodes*.
- If *reductio-supposition-nodes* was previously empty, then for every c-list *cl* in processed-conclusions, if (*reductio-interests cl*) is T, then for every c-list-node *N*, GENERATE-REDUCTIO-INTERESTS for *N*.
- If the value of *reductio-interests* for the node-c-list of *node* is T (i.e., *node* is syntactically appropriate), GENERATE-REDUCTIO-INTERESTS for *node*.

ADOPT-REDUCTIO-INTEREST *node*.

If the list of *reductio-supposition-nodes* is nonempty, *node* is not a reductio-supposition, *node* is syntactically appropriate, and either *node* was not inferred by discharging an interest-link or the resultant-interest of that link (which *node* satisfies) is not a cancelled-interest, then GENERATE-REDUCTIO-INTERESTS for *node*.

GENERATE-REDUCTIO-INTERESTS *node*.

- Let *P* be the negation of the node-formula of *node*, let *sup* be the non-reductio-supposition of *node*, and let *sequent* be the sequent constructed from this supposition and formula.
- If there is no previously existing interest in *sequent*, construct one, setting the value of its slot *direct-reductio-interest* to be the unit set of *node*.
- If there is already such an interest *interest:*
  - add *node* to the list *direct-reductio-interest* for *interest*.
  - if *interest* was not previously a reductio-interest, reconstrue it as one;
  - recompute the degree-of-interest and the interest-priority for *interest*.

The search for a previously existing interest in *sequent* is facillitated by appealing to the c-list-contradictor of the c-list of *node*. Any such interest must be in the i-list-interests of the corresponding-i-list of that c-list.

Direct-reductio-interests are used for making reductio-inferences. This is done by DISCHARGE-REDUCTIOS, and will be discussed shortly. Other reductio-interests are adopted by reasoning backwards from direct-reductio-interests, and as such are discharged in the normal way by DISCHARGE-INTEREST-IN and DISCHARGE-LINK. There is a complication, however, resulting from the schematic character of the interest-supposition in reductio-interests. In chapter three, the rule for interest-discharge required that the node-supposition of the inference-node be a subset of the interest-supposition of the interest it discharges. Using schematic reductio-interests, that condition is too strong. This is for two reasons. First, the reductio-suppositions are not listed in the interest-supposition, so the most we could require is that the non-reductio-supposition of the inference-node be a subset of the interest-supposition. However, even that is too strong.

Consider the following reasoning:

```
conclusions                               interests

R
                                          (P -> Q)
suppose P      (for conditionalization)
                                          Q/{P}
suppose ~Q   (reductio-supposition)
                                          ~R  (reductio-interest)
                                            .
                                            .  (interest-link)
                                            .
                                            S
```

Suppose further reasoning produces the following conclusion:

S/{P,~Q}

Then the reasoner should discharge the interest-link to infer:

~R/{P,~Q}
Q/{P}   (by reductio)
(P -> Q) (by conditionalization)

The thing to observe here is that although we want the conclusion S/{P,~Q} to discharge the interest S, the non-reductio-supposition of the conclusion is {P}, which is not a subset of the interest-supposition (the latter being empty). The source of the difficulty is that direct-reductio-interests are no longer tied by interest-links to the interests generating them. Instead, we regard each direct-reductio-interest as being generated by *all* interests that are syntactically appropriate for the generation of reductio-suppositions. Accordingly, the direct-reductio-interest (in this case, ~R) cannot inherit the supposition of the generating interest in Q/{P}.

Let us define the *inherited-non-reductio-supposition* of an interest to be the list of all non-reductio-suppositions not in the interest-supposition that can be reached by working backwards from the interest through right-links, generating-nodes, and generating-interests. What should be required for interest-discharge of a reductio-interest is that the non-reductio-supposition of the discharging node be a subset of the union of the interest-supposition and its inherited-non-reductio-supposition.

This can be simplified by noting that all reductio-interests have the same inherited-non-reductio-supposition. This is because the list of generating-nodes of a direct-reductio-interest is taken to be the list of all reductio-suppositions. Thus we can simply compute a single list of *inherited-non-reductio-suppositions* as we go along, augmenting it as needed when new reductio-suppositions are made. Then what is required for interest-discharge is that the supposition of the inference-node be "appropriately-related" to the supposition of the interest, taking the resultant-interest of the interest-link to be the target, in the following sense:

APPROPRIATELY-RELATED-SUPPOSITIONS *node interest optional argument: target*

- If *target* is not supplied, let it be *interest*.
- If *target* is not a reductio-interest, then the node-supposition is a subset of the interest-supposition of *interest*.
- Otherwise, every member of the non-reductio-supposition of *node* is contained in either the interest-supposition of *interest* or in the list *inherited-non-reductio-suppositions*.

This revision affects the application of DISCHARGE-INTEREST-IN, DISCHARGE-LINK, and MAKE-INFERENCE-FROM-LINK.

When *node* is a new inference-node produced by either DRAW-CONCLUSION or DRAW-REDUCTIO-CONCLUSION, we check to see whether we can draw a reductio-conclusion from it. For this purpose we check to see whether *node* contradicts an earlier conclusion.

If so, we can infer the negation of a reductio-ancestor of the contradiction. In the latter connection, there is one qualification. We can distinguish between "base-reductio-suppositions" and other reductio-suppositions. A base-reductio-supposition is one that is generated from interests that are not themselves reductio-interests. Reductio-suppositions that are not base-reductio-suppositions can be generated from each other in any order, and so they can be discharged in any order. But base-reductio-suppositions must always be discharged last, after all other reductio-suppositions have been discharged. In the following, *i-list* is the corresponding-i-list of the node-c-list of *node*:

DISCHARGE-REDUCTIOS *node i-list*

- DISCHARGE-FORTUITOUS-REDUCTIOS *node*
- If *node* has reductio-ancestors, then for any direct-reductio-interest *interest* whose interest-formula is the same as *node's* node-formula (i.e., *interest* is chosen from *i-list*) and whose interest-supposition is appropriately-related to that of *node*:
  - Let Y be the node-supposition of *node*.
  - For each *node** in the direct-reductio-interest slot of *interest* (i.e., *interest* is a reductio-interest in the negation of the node-sequent of *node**):
    - Let Y^* be the node-supposition of *node**.
    - For each reductio-ancestor R of *node* or *node** , if either R is not a base-reductio-supposition or it is the only reductio-ancestor of *node* or *node** , and the node-formula of R is P, then DRAW-REDUCTIO-CONCLUSION *P node node* R Y Y* interest*.

DRAW-REDUCTIO-CONCLUSION *P node node* R Y Y* interest*

If neither *node* nor *node** is a cancelled-node, and $\neg P$ is not in Y or Y^*, then let *sup* be $Y \cup Y^*$-$\{P\}$, let S be the sequent $\langle sup, \neg P \rangle$, and let *NDA* be the union of the crossproduct of the nearest-defeasible-ancestors of *node* and the nearest-defeasible-ancestors of *node**. If not SUBSUMED *S NDA*:

- If there is an inference-node of kind "inference" supporting S, let *conclusion* be that inference-node. Append *NDA* to the list of nearest-defeasible-ancestors for *conclusion*.
- Otherwise, construct a new inference-node *conclusion* supporting S, setting its deductive-only slot to be NIL.
- Build a support-link *link* recording the inference of *conclusion* from {*node*,*node**} in accordance with *reductio*, construct the set of new non-circular arguments this produces for *conclusion* and its inference-descendants, and recompute the lists of nearest-defeasible-ancestors for the inference-descendants.
- If the preceding step does produce new arguments for *conclusion*:
  - Record *link* as a support-link for *conclusion*.
  - If *conclusion* is newly constructed, add it to the *inference-graph* and store it in the list of *conclusion*, and make *NDA* to the list of nearest-defeasible-ancestors for *conclusion*
  - When R is non-NIL and *conclusion* is a deductive-node, add the generating-interests of R to the list of enabling-interests of *conclusion*.
  - If *conclusion* is a deductive-node, and *interest* is not NIL, add *interest* to the list of enabling-interests for *conclusion*.
  - CANCEL-SUBSUMED-LINKS *link*.
  - If *conclusion* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, let *old-degree* be the old maximal-degree of support for *conclusion*. Let *i-list* be the corresponding i-list of the conclusion-c-list of *conclusion*.
    - If *conclusion* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, DISCHARGE-INTEREST-IN-DEFEATERS *conclusion i-list old-degree*
    - Add *ink* to the list *new-links*.
    - If *conclusion* already existed but this inference increases the maximal-degree-of-support of *node*, ADJUST-SUPPORT-FOR-CONSEQUENCES *con-*

clusion old-degree

- If *conclusion* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, DISCHARGE-INTEREST-IN *conclusion i-list old-degree interest*
- If *conclusion* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, DISCHARGE-REDUCTIOS *conclusion i-list old-degree interest*.
- Set the old-undefeated-degree-of-support of *node* to be the undefeated-degree-of-support of *node*.

When we adopt a new reductio-interest *interest*, it may be possible to discharge that interest immediately. This is done by DISCHARGE-RETROSPECTIVE-REDUCTIOS.

DISCHARGE-RETROSPECTIVE-REDUCTIOS *node interest*

For each node *N* supporting the interest-formula of *interest*, if the maximal-degree-of-support of *N* is greater than or equal to the degree-of-interest of *interest*, and *N* and *interest* have appropriately-related-suppositions, let *Y* be the node-supposition of *N*, and let *Y** be the node-supposition of *node.*

- For each reductio-ancestor *R* of *N*, if either *R* is not a base-reductio-supposition or it is the only reductio-ancestor of *N*, and the node-formula of *R* is *P*, then DRAW-REDUCTIO-CONCLUSION *P N node R Y Y* interest*.

Generating-reductio-interests can convert prior interests that were not reductio-interests into reductio-interests. When that happens, we must check whether they can be immediately discharged. This is done by DISCHARGE-NEW-REDUCTIO-INTEREST, which is analogous to DISCHARGE-RETROSPECTIVE-REDUCTIOS.

In addition, the changes indicated in bold-face are made to pre-existing rules:

REASON-BACKWARDS-FROM *interest priority*

- Let *sequent* be the interest-sequent of *interest*, δ its degree-of-interest, and δ* its last-processed-degree-of-interest.
- For each backwards-reason *R* of strength at least δ but less than δ*, if *interest* is still uncancelled and the (first) conclusion of *R* matches the interest-formula of *interest* by pattern-matching using the reason-variables, and *binding* is the resulting assignment of values to the reason-variables, apply MAKE-BACKWARDS-INFERENCE to *R*, *binding*, and *interest*.
- **If the use of *reductio-ad-absurdum* is enabled, apply DISCHARGE-RETROSPECTIVE-REDUCTIOS and make a reductio-supposition for the purpose of getting *interest*. (This will be explained in chapter four.)**
- For each auxiliary rule for backwards reasoning *R*, apply *R* to *S*.

REASON-FORWARDS-FROM *node*

- DISCHARGE-INTEREST-SCHEMES *node* .
- APPLY-FORWARDS-REASONS *node*.
- If (interests-discharged? *node*) is NIL, apply DISCHARGE-INTEREST-IN to *node* and the corresponding-i-list of the node-c-list of *node.*
- **If *reductio-ad-absurdum* reasoning is enabled, apply ADOPT-REDUCTIO-INTERESTS and DISCHARGE-DELAYED-REDUCTIOS to *node*.**
- For each auxiliary rule *R* for forwards reasoning, apply *R* to *node*.

APPLY-FORWARDS-REASONS *node*

For each forwards-reason *R*, if **either *R* is not defeasible or *node* is not deductive-only** and there is a list *nodes* such that (1) *node* is in *nodes*, (2) the other members of *nodes* are processed conclusions, (3) the members of *nodes* satisfy the applicability-conditions of *R*, and (4) where *B* is the list of node-formulas of the members of *nodes*, *B* matches the forwards-premises of *R* by pattern-matching, and *binding* is the resulting assignment to the reason-variables, then MAKE-FORWARDS-INFERENCE *R binding basis*.

DRAW-CONCLUSION *P B R discharge (optional-argument: interest)*

If no member of *B* is cancelled, let *sup* be the union of the node-suppositions of the members of *B*, less *discharge*, and let *S* be the sequent ⟨*sup,P*⟩. If *S* has not been validated deductively:

- If *R* is not defeasible, let *NDA* be the set of unions of the crossproducts of the sets of nearest-defeasible-ancestors of members of *B**, and otherwise let *NDA* be NIL.
- If either *R* is defeasible or not SUBSUMED *S NDA* :
    - **Let *deductive-only* be T if *R* is not *reductio* and some member of *B* is a deductive-only conclusion; otherwise let *deductive-only* be NIL.**
    - If there is an inference-node of kind :inference supporting *S*, let *node* be that inference-node. **If *node* is a deductive-only conclusion, but *deductive-only* is NIL, redefine the *deductive-only* slot in *node* to make it NIL.**
    - Otherwise, construct a new inference-node *node* supporting *S*. **Make *node* deductive-only iff *deductive-only* is T.**
    - If *node* is a deductive-node, and *interest* is not NIL, add *interest* to the list of enabling-interests for *node*.
    - Build a support-link *link* recording the inference of *node* from *B* in accordance with *R*, and recompute the lists of node-ancestors and nearest-defeasible-ancestors for the inference-descendants.
    - If the preceding step does produce new arguments for *node*:
        - ADOPT-INTEREST-IN-DEFEATERS-FOR *link*.
        - Record *link* as a support-link for *node*.
        - If *node* is newly constructed, add it to the *inference-graph* and store it in the list of *conclusions*
        - If *R* is defeasible, add {*node*} to the list of nearest-defeasible-ancestors of *node*; otherwise append *NDA* to the list of nearest-defeasible-ancestors of *node*.
        - CANCEL-SUBSUMED-LINKS *link*.
        - If *node* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, let *old-degree* be the old maximal-degree of support for *node*:
            - Let *i-list* be the corresponding i-list of the conclusion-c-list of *node*.
                - If *node* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, DISCHARGE-INTEREST-IN-DEFEATERS *node i-list old-degree*
                - Add *ink* to the list *new-links*.
                - If *node* already existed but this inference increases the maximal-degree-of-support of *node*, ADJUST-SUPPORT-FOR-CONSEQUENCES *node old-degree*
                - If *node* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, DISCHARGE-INTEREST-IN *node i-list old-degree interest*
                - **If *node* is either newly-constructed, or its maximal-degree-of-support has increased as a result of adding this support-link, and *reductio-ad-absurdum* is enabled, DISCHARGE-REDUCTIOS *node i-list old-degree interest*.**

ADJUST-SUPPORT-FOR-CONSEQUENCES *node old-degree*.

- For all nodes *N* that are support-link-targets of consequent-links of *node*, if the maximal-degree-of-support of *N* is old-degree, let *N-old* be the maximal-degree-of-support for *N*, recompute the maximal-degree-of-support for *N*, and if it increases, let *i-list* be the corresponding-i-list of the node-c-list of *N*. If there is such an i-list:
    - DISCHARGE-INTEREST-IN-DEFEATERS *N i-lists old-degree*
    - DISCHARGE-INTEREST-IN *N i-lists old-degree*
    - if *N* is a processed-conclusion, DISCHARGE-INTEREST-SCHEMES *N old-degree*

- **if *reductio-ad-absurdum* is enabled, DISCHARGE-REDUCTIOS *node i-list old-degree.***
- ADJUST-SUPPORT-FOR-CONSEQUENCES *N N-old*.

DISCHARGE-LINK *link* δ (using *priority*, if available, to simplify the computation of interest-priorities):

- For every inference-node *N* in the c-list-nodes of the correspondingc-list of the interest-i-list of the resultant-interest of *link*, if **either the link-rule of *link* is a deductive rule or *N* is not deductive-only**, *N* satisfies the discharge-condition (if any) of the resultant-interest and the node-supposition of *N* is a subset of the interest-supposition of the resultant-interest:
  - If *link* is an answer-link, add *N* to the list of supporting-nodes of *link*□ and add *N* to the list of query-answers for the resultant-interest of *link*.
  - If *link* is not an answer-link and has no remaining-premises, where *discharge* is the link-discharge of *link* and *R* is the link-rule, let *formula* be the interest-formula of the resultant-interest of *link*, and DRAW-CONCLUSION *formula supporting-nodes R discharge*.
  - If instead *link* has remaining-premises, construct a new interest-link by inserting the validating node into the list of supporting-nodes of the discharged link, deleting the first remaining-premise from the list of remaining-premises and making it the link-interest. Apply DISCHARGE-LINK to the new-link and δ.
  - If *N* is a deductive-node, CANCEL-INTEREST-IN *interest*.
  - If *N* is not a deductive-node, then for all generated-suppositons of *interest* that have no other generating-interests, if *N* is deductive-in the supposition, apply CANCEL-NODE to the supposition.

OSCAR incorporates a further refinement of these rules. There is a choice to be made regarding when DISCHARGE-REDUCTIO is to be applied. It could be applied either when a new conclusion is first drawn (by building this into DRAW-CONCLUSION and DRAW-REDUCTIO-CONCLUSION), or it could be applied when the conclusion is retrieved from the *inference-queue*. For reasoning without quantifiers, the former strategy will be employed. Experiment indicates that in first-order logic, a mixture of these two strategies is actually the best. This will be discussed in chapter five.

These rules have the consequence that reductio-suppositions do not figure into the formulation of reductio-interests, thus avoiding the spiraling of interests described above, but when reductio-interests are discharged, the appropriate suppositions are computed so that they include any reductio-suppositions upon which the inference depends.

As described, the rules for discharging direct-reductio-interests only license inferences to the negations of reductio-ancestors of the contradictory conclusions giving rise to the inference. Although this is very natural, the resulting system is not complete for the propositional calculus. The difficulty is that it only draws reductio-conclusions from inconsistent suppositions. If the inconsistency arises from inconsistent premises, no reductio-conclusion can be drawn. The following is a simple example:

```
Given premises:
   P    justification = 1
   ~P    justification = 1
Ultimate epistemic interests:
   Q    interest = 1

 # 1
 conclusion 1: P
 given
 Non-reductio-supposition: nil
 # 2
 conclusion 2: ~P
 given
 Non-reductio-supposition: nil
```

```
---------------------------------------------------------------------------
Retrieving #<Query #1: Q> from the inference-queue.
                                  # 1
                                  interest: Q
                                  This is of ultimate interest
  # 3
  conclusion 3: ~Q    supposition: { ~Q }
  reductio-supposition
  Non-reductio-supposition: nil
---------------------------------------------------------------------------
Retrieving #<Conclusion #1:  P> from the inference-queue.
                                  # 2
                                  reductio interest: ~P
                                  using conclusion 1
---------------------------------------------------------------------------
Retrieving #<Conclusion #2:  ~P> from the inference-queue.
.........................................................................................................................................
Cancelling  #<Interest 2: ~P supposing {  }>
.........................................................................................................................................
                                  # 3
                                  reductio interest: P
                                  using conclusion 2
---------------------------------------------------------------------------
Retrieving #<Interest 3: P supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Conclusion #3:  ~Q supposing { ~Q }> from the inference-queue.
                              Readopting interest in:
                                  # 1
                                  reductio interest: Q
                                  This is of ultimate interest


================= ULTIMATE EPISTEMIC INTERESTS ==================
  Interest in Q
  is unsatisfied.  NO ARGUMENT WAS FOUND.
---------------------------------------------------
```

For the purposes of modeling human reasoning, I am not convinced that this is an untoward result. However, for some purposes we do want completeness for the propositional calculus. To achieve that, a call to DISCHARGE-FORTUITOUS-REDUCTIOS has been added to the reductio-discharge rules:

DISCHARGE-FORTUITOUS-REDUCTIOS *node*.

For each *node** in the list of c-list-nodes for the c-list-contradictor of the c-list of *node*:

- If *node* and *node** both have empty node-suppositions, then for each ultimate-epistemic-interest, infer the query-formula from *node* and *node**.
- If *node* discharges some interest generated by a supposition in the list of non-reductio-suppositions of *node**, and the node-supposition of *node* is a subset of the node-supposition of *node**, then for each supposition *sup* in the list of non-reductio-suppositions of *node**, if *node** is deductive-in *sup*, then for every generated-interest *in* of *sup*, if *node* and *in* have appropriately-related-suppositions, then where *P* is the interest-formula of *in*, DRAW-CONCLUSION *P* {*node*, *node**} "fortuitous-reductio" NIL.
- If *node** discharges some interest generated by a supposition in the list of non-reductio-suppositions of *node*, and the node-supposition of *node** is a subset of the node-supposition of *node*, then for each supposition *sup* in the list of non-reductio-suppositions of *node*, if *node* is deductive-in *sup*, then for every generated-interest *in* of *sup*, if *node** and *in* have appropriately-related-suppositions, then where *P* is the interest-formula of *in*, DRAW-CONCLUSION *P* {*node*, *node*} "fortuitous-reductio" NIL.

This governs "fortuitous reductio", which licenses the drawing of reductio-conclusions from any inconsistent pairs of deductive conclusions. With this addition, the above problem is done as follows:

```
Given premises:
    P    justification = 1
    ~P    justification = 1
Ultimate epistemic interests:
    Q    interest = 1

  # 1  P
  given
  # 2   ~P
  given
---------------------------------------------------------------------------
Retrieving #<Query #1: Q> from the inference-queue.
                                # 1
                                interest: Q
                                This is of ultimate interest
  # 3   ~Q    supposition: { ~Q }
  reductio-supposition
---------------------------------------------------------------------------
Retrieving #<Conclusion #1:  P> from the inference-queue.
                                # 2
                                reductio interest: ~P
                                using node 1
---------------------------------------------------------------------------
Retrieving #<Conclusion #2:  ~P> from the inference-queue.
.....................................................................................................................................
Cancelling  #<Interest 2: ~P supposing {  }>
.....................................................................................................................................
                                # 3
                                reductio interest: P
                                using node 2
---------------------------------------------------------------------------
Retrieving #<Interest 3: P supposing {  }> from the inference-queue.

Node #1 discharges reductio-interest #1, generated by node #2
  # 4   Q
  Inferred by support-link #1 from { 1 , 2 } by fortuitous-reductio

           ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in Q
  is answered affirmatively by conclusion 4
---------------------------------------------------
```

The cancellation rules must be changed to accommodate reductio-interests. A reductio-interest can be cancelled normally by finding a deductive argument for a conclusion that subsumes it. However it must be borne in mind that direct-reductio-interests do not work via interest-links. This creates a minor problem, because a direct-reductio-interest might also be of interest for some other reason (e.g., conditionalization), and hence have right-links. Cancelling the resultant-interest of those right-links should not cancel the interest. A mechanism must be adopted to handle this. A further change required is that if an interest is cancelled without being discharged, and it led to the making of a reductio-supposition, the node recording that reductio-supposition should be cancelled and its inference-descendants should be cancelled. To implement this, we have a slot in interests recording "generated-reductio-suppositions". Furthermore, cancelling a node should cancel interest in any reductio-interests generated by it. To implement all of this, we have a slot in inference-nodes recording "generated-interests" and "generating-interests", and a slot in interests recording "generating-nodes" and "generated-suppositions". Interests generate nodes recording suppositions made for reductio reasoning and conditionalization, and those suppositions in turn generate new interests. Inference nodes also generate new nodes by inference, and interests generate each other by backwards reasoning (recorded in interest-links, discharged or otherwise). As a first approximation, when we cancel an interest, we should cancel all interests and nodes generated from it provided they are not generated in any other way. An immediate qualification is required, in that among the nodes generated by an interest being cancelled as a result of being discharged will be the node discharging it. We do not want to cancel that node — just the nodes leading up to it.

A major complication arises from the way the reductio rules treat inferences as having "schematic" suppositions. This can lead to generation cycles. For instance, interest in *Q* might lead to interest in *P* by backwards reasoning. Interest in *P* might lead to the reductio-supposition ~*P*, and reductio-interest in *P* relative to the supposition {~*P*}. However, the supposition {~*P*} will not be recorded as part of the interest-supposition of the reductio-interest, so the reasoner simply readopts interest in *P*. This is not circular reasoning, because the single interest is a schematic interest that really encodes two different interests. However, if we look at the generation paths for this interest, they are circular, as diagrammed in figure 1. This creates a problem for the cancellation rules, because cancelling interest in *Q* should cancel the interest *P* and the supposition ~*P*; however this will not be accomplished by the simple cancellation rules just described, because that interest is generated in another way, viz., from the reductio-supposition ~*P*. Proper cancellation rules must be able to compute such "unanchored" circularities and cancel them.

The most straightforward algorithm for detecting such circularities would search backwards from a node or interest along generation-paths, rejecting each path as it becomes circular, until either a noncircular path is found or all paths are rejected. But that algorithm turns out to be horribly inefficient when applied to large problems. The algorithm used by OSCAR works as follows. First, when a node or interest is cancelled, the set of all "dependencies" is computed. These are interests and nodes occurring on some generation-path from the cancelled node or interest, where these nodes and interests are defined recursively to include the node-consequences of a node, the link-interests of left-links (cancelled or not), suppositions generated by interests, and interests generated by reductio-suppositions (these are the direct-reductio-interests). Second, we compute the subset of dependencies that are "directly-anchored" by also being generated by interests or reductio-suppositions different from the initial one and not contained in the set of dependencies. We recursively delete the directly-anchored nodes and interests and everything generated from them from the set of dependencies. The remaining nodes and interests are unanchored and can be cancelled.

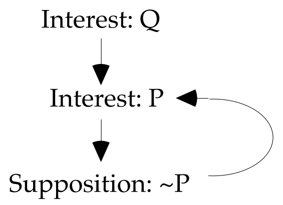


Figure 1. Circular generation paths.

To implement the new cancellation rules, two slots are added to inference-nodes: *enabling-interests* and *generated-direct-reductio-interests*:

The **enabling-interests** of a node are the resultant-interests of interest-links such that the node is inferred by discharging the interest-link.

The **generated-direct-reductio-interests** of a node are the interests containing the node in their list *direct-reductio-interest*.

In addition, two slots are added to interests: *discharging-nodes* and *cancelling-node*.

> If a node answers a query, it is inserted into the list of **discharging-nodes** for the interest recording the query. When an inference is made from an interest-link, the nodes from which the inference is made are inserted into the lists of discharging-nodes of the corresponding members of the link-interests. When a reductio-conclusion is drawn by finding a node that discharges a direct-reductio-interest interest, the node is inserted into the list of discharging-nodes for the interest. In other words, the discharging-nodes of an interest are the nodes that discharge the interest in such a way that an interest from which the discharged interest was derived (by backwards reasoning or by generating a reductio-supposition and then adopting the discharged interest as a reductio-interest) can then be be discharged.
>
> When an interest is cancelled as a result of inferring an appropriate node, that node is the **cancelling-node** of the interest.

(CANCEL-INTEREST-IN *interest*) begins by computing the lists of dependent-nodes and dependent-interests. These lists are constructed recursively according to the following rules:

- *interest* is inserted into the list of dependent-interests
- suppositions generated by dependent-interests are dependent-nodes
- if *L* is a left-link of a dependent-interest and *in* is a link-interest or cancelled-link-interest of *L*:
  - *in* is a dependent-interest
  - any discharging-nodes of *in* are dependent-nodes
- node-consequences of a dependent-node are dependent-nodes
- direct-reductio-interests in the negation of a dependent-node (i.e., the node is in the list *direct-reductio-interest* for the interest) are dependent-interests
- if a reductio-supposition is in the list of dependent-nodes, and all direct-reductio-interests are put in the list of dependent-interests.

Then nodes and interests that can be generated from nodes and interests not in the lists of dependent-nodes and interests are removed from the lists of dependent-nodes and interests. Such nodes and interests are said to be "anchored".

In computing the anchored nodes and interests, we first compute the "directly-anchored" nodes and interests. Let *independent-reductio-suppositions* be the set of reductio-supposition-nodes that are not in the list of dependent-nodes. The directly-anchored-nodes are those dependent-nodes that either (1) have an uncancelled generating-interest that is not a dependent-interest, (2) have a support-link whose basis consists of nodes that are uncancelled and not dependent, (3) were obtained by discharging a direct-reductio-interest in the negation of a node that is neither cancelled nor dependent and *independent-reductio-suppositions* is nonempty, or (4) have a discharged-interest with a right-link whose resultant-interest is neither cancelled nor in the list of dependent-interests. Condition (4) means that if we are "finished with" a node that is obtained by making an inference from a link (perhaps because the discharged-interest is just one of several interests that must be discharged to make an inference from a right-link of the interest), then the node will not be cancelled. The directly-anchored interests are those dependent-interests other than *interest* that either (1) have generating-nodes that are not dependent-nodes, (2) are direct-reductio-interests and (a) *independent-reductio-suppositions* is nonempty and (b) some node in their *direct-reductio-interest* slot is not a dependent-node, or (3) have a right-link whose resultant-interest is not a dependent-interest.

We remove the directly-anchored nodes and interests from the lists of dependent-nodes and interests. Then we recursively anchor nodes and interests that can be derived from them, removing them from the lists of dependent-nodes and interests as we go. The rules for this are as follows, where *interest* is the interest with which interest-cancellation began:

- If *in* is an anchored interest:
  - for every left-link *L* of *in*:
    - anchor the link-interests of *L* that are different from *interest*

        • for every cancelled-link-interest *in** of *L* that is different from *interest*:
            • anchor *in**
            • if *cn* is the cancelling-node of *in**, anchor *cn*
    • anchor every generated-supposition of *in**
• if *node* is an anchored node:
    • anchor any node-consequences of *node* having a support-link whose basis consists of uncancelled non-dependent-nodes
    • if *node* is a reductio-supposition:
        • return *node* to the list of *independent-reductio-suppositions*
        • for every direct-reductio-interest *in*:
            • if *in* is different from *interest* and contains some uncancelled non-dependent-nodes in its *direct-reductio-interest* slot, anchor *in*
            • anchor any cancelling-node of *in*
    • for every generated-direct-reductio-interest *in* of *node*, if *independent-reductio-suppositions* is nonempty:
        • if *in* is different from interest, anchor *in*.
        • anchor any cancelling-node of *in*

After all the anchored nodes and interests are removed from the lists of dependent-nodes and interests, those that remain can be cancelled.

CANCEL-NODE works similarly. These computations can be displayed by setting *d-trace* to T (the default is NIL). The following is an example of the display produced:

```
COMPUTING DEPENDENCIES from #<Interest 2: ((~q -> ~p) -> (p -> q)) supposing {  }>
|generated-supposition: #<Node 1>
|. node-consequence: #<Node 4>
|. . node-consequence: #<Node 5>
|. . . node-consequence: #<Node 6>
|. generated-direct-reductio-interest: #<Interest 5: ~(~q -> ~p) supposing { (~q -> ~p) }>
|left-link: #<Interest 3: (p -> q) supposing { (~q -> ~p) }>
|. generated-supposition: #<Node 2>
|. left-link: #<Interest 4: q supposing { p , (~q -> ~p) }>
|. . generated-supposition: #<Node 3>
|. . . generated-direct-reductio-interest: #<Interest 6: q supposing {  }>
ANCHORING-NODES AND INTERESTS
Node 6 is directly anchored by being the cancelling-node for interest 2 which has a right-link to interest 1
Directly-anchored node: #<Node 6>
. . . . . . . . . Cancelling  #<Node 3>
. . . . . . . . . Cancelling  #<Node 2>
. . . . . . . . . Cancelling  #<Node 5>
. . . . . . . . . Cancelling  #<Node 4>
. . . . . . . . . Cancelling  #<Node 1>
. . . . . . . . . Cancelling  #<Interest 6: q supposing {  }>
. . . . . . . . . Cancelling  #<Interest 4: q supposing { p , (~q -> ~p) }>
. . . . . . . . . Cancelling  #<Interest 3: (p -> q) supposing { (~q -> ~p) }>
. . . . . . . . . Cancelling  #<Interest 5: ~(~q -> ~p) supposing { (~q -> ~p) }>
. . . . . . . . . Cancelling  #<Interest 2: ((~q -> ~p) -> (p -> q)) supposing {  }>
```

The prioritization rules must also be changed, for two reasons. First, the interest-priority of a reductio-interest should be determined by the interest-priorities of the interests from which it is derived by backwards reasoning, but that can no longer be computed by looking at its right-links because direct-reductio-interests no longer have right-links. Second, the complexity of reductio-suppositions is considerably reduced because the node-suppositions record only the formula that is newly-supposed, and this will have the affect of inflating the priority of reasoning from such suppositions. These problems can be handled simultaneously by letting the discounted-node-strength of a supposition be determined by the interest-priority of its generating interests, and letting the interest-priority of a direct-reductio-interest be determined by the discounted-node-strengths of the suppositions generating it. More precisely, the following rules will be adopted:

Inference-nodes

- If a node is defeated, its discounted-node-strength is α_0.
- If an undefeated node has an empty list of generating-interests, its discounted-node-strength is the the maximum (over its node-arguments) of the product of the discount-factor of the support-link-rule of the last support-link in the argu-

ment and the strength of the argument. (This case includes all non-suppositional nodes.)
- If a node is a supposition, its discounted-node-strength is the maximum of:
  (1) the product of **reductio-discount** and the maximum of the interest-priorities of the generating-interests for which it is a reductio-supposition; and
  (2) the interest-priorities of the generating-interests for which it is not a reductio-supposition.

Queries
- The interest-priority of a query from *permanent-ultimate-epistemic-interests* is its degree-of-interest.
- The interest-priority of any other query is its degree-of-interest if it has not been answered yet, and it is the product of **answered-discount** and its degree of interest if it has already been answered.

Interests
- If an interest has neither generating-nodes nor undefeated right-links nor undefeated discharged-right-links, its interest-priority is α_0. (This includes interest in defeaters.)
- Otherwise, the interest-priority is the maximum of:
  (1) the discounted-node-strengths of its generating-nodes that are not reductio-suppositions;
  (2) the product of **reductio-interest** and the maximum of the discounted-node-strengths of its generating-nodes that are reductio-suppositions;
  (3) for each of its right-links or discharged-right-links, the product of the discount-factor of the link-rule and the interest-priority of the resultant-interest.

The computation and recomputation of discounted-node-strengths and interest-priorities is made more complicated by the same generation-loops that complicated the cancellation rules. QUEUE-PERCEPT, QUEUE-PREMISE, and QUEUE-DESIRE set the discounted-node-strength to the same thing as the maximal-degree-of-support. QUEUE-SUPPOSITION sets the discounted-node-strength to the interest-priority of the generating-interest, and MAKE-REDUCTIO-SUPPOSITION sets the discounted-node-strength to the product of **reductio-discount** and the interest-priority of the generating-interest. UPDATE-BELIEFS calls RECOMPUTE-PRIORITIES. This begins with the lists of new-beliefs, new-retractions, altered-interests, and altered-queries. These are the nodes whose undefeated-degrees-of-support have changed, the interests having right-links whose defeat-statuses have changed, and the queries whose answered-status has changed. From these we can compute the list of affected-nodes and affected-interests, which are all of those supposition-nodes and interests hereditarily generated from the altered-nodes and altered-links. We then proceed as follows:

Repeat the following until the lists of altered-nodes and altered-interests are empty:
- For each altered-node or altered-interest whose discounted-node-strength or interest-priority can be computed without appealing to any other altered-nodes or altered-interests, we do so and remove them from the lists of altered-nodes and altered-interests. Repeat this step until no further nodes or interests can be removed. If there are no generation-cycles, this will get them all, but if there are cycles, some may remain.
- For any remaining nodes or interests, we want to compute their discounted-nodes-strengths and interest-priorities just in terms of the nodes and interests not involved in the cycles. Cycles arise from direct-reductio-interests that also have other sources and reductio-suppositions that are also non-reductio-suppositions. So for any such interests and suppositions, compute their interest-priorities and discounted-node-strengths just in terms of those of their sources that are no longer contained in the lists of altered-nodes or altered-interests.

With these changes, OSCAR handles reductio reasoning properly without replicating backwards reasoning. For instance, problem #29 could now be done as follows:[1]

```
Problem #29

Given premises:
Ultimate epistemic interests:
     (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))    interest = 1.0

                                        # 1
                                        interest: (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
                                        This is of ultimate interest

  # 1   ((p v q) & ((~p v q) & (p v ~q)))    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  supposition
                                        # 2
                                        interest: ~(~p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 1 by conditionalization
---------------------------------------------------------------------------
1:    Retrieving #<Node 1> from the inference-queue.  Preference = 1.0

  # 2   (p v q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #1 from { 1 } by simp
  undefeated-degree-of-support = 1.0

  # 3   ((~p v q) & (p v ~q))    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #2 from { 1 } by simp
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
2:    Retrieving #<Node 2> from the inference-queue.  Preference = 0.3333

  # 4   (~p -> q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #3 from { 2 } by disj-simp
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
3:    Retrieving #<Node 4> from the inference-queue.  Preference = 0.25
---------------------------------------------------------------------------
4:    Retrieving #<Interest 2: ~(~p v ~q) supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the
inference-queue.  Preference = 0.1666
                                        # 3
                                        interest: p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                        For interest 2 by i-neg-disj
---------------------------------------------------------------------------
5:    Retrieving #<Interest 3: p supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-queue.
Preference = 1.0

  # 5   ~p    supposition: { ~p }
  supposition
                                        # 4
                                        reductio interest: ~(~p -> q)    supposition: {((p v q) & ((~p v q) & (p v ~q))) }
                                        using node 4
---------------------------------------------------------------------------
6:    Retrieving #<Node 5> from the inference-queue.  Preference = 0.25
                                        # 5
                                        reductio interest: ~q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                        using node 6

  # 6   q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~p }
  Inferred by support-link #4 from { 4 , 5 } by modus-ponens2
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
7:    Retrieving #<Node 6> from the inference-queue.  Preference = 1.0
---------------------------------------------------------------------------
```

[1] This example is contrived. The prioritization rules have been changed, so OSCAR actually does this problem in a different way now.

```
8:    Retrieving #<Interest 5: p supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-
queue.  Preference = 0.25
------------------------------------------------------------------------
9:    Retrieving #<Interest 5: ~q supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-
queue.  Preference = 0.125

  # 7   q    supposition: { q }
  supposition
------------------------------------------------------------------------
10:    Retrieving #<Node 7> from the inference-queue.  Preference = 0.25
------------------------------------------------------------------------
11:    Retrieving #<Node 3> from the inference-queue.  Preference = 0.1111

  # 8   (~p v q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #5 from { 3 } by simp
  undefeated-degree-of-support = 1.0

  # 9   (p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #6 from { 3 } by simp
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
12:    Retrieving #<Node 8> from the inference-queue.  Preference = 0.25
                                    # 6
                                    reductio interest: ~(p -> q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                    using node 10

  # 10   (p -> q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #7 from { 8 } by disj-simp
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
13:    Retrieving #<Node 10> from the inference-queue.  Preference = 0.3333
------------------------------------------------------------------------
14:    Retrieving #<Node 9> from the inference-queue.  Preference = 0.25

  # 11   (~p -> ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #8 from { 9 } by disj-simp
  undefeated-degree-of-support = 1.0
------------------------------------------------------------------------
15:    Retrieving #<Node 11> from the inference-queue.  Preference = 0.2
  Node 12 discharges interest 7
..................................................................................................................................
Cancelling interest 7
. Cancelling  #<Interest 7: ~q supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
..................................................................................................................................
  Node 12 discharges interest 5
  Node 13 discharges interest 8
..................................................................................................................................
Cancelling interest 8
. Cancelling  #<Interest 8: p supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
..................................................................................................................................
  Node 13 discharges interest 5
..................................................................................................................................
Cancelling interest 5
. Cancelling  #<Interest 5: p supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
..................................................................................................................................
  Node 13 discharges interest 3
                                    # 7
                                    interest: q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
                                    For interest 2 by i-neg-disj using node 13
..................................................................................................................................
Cancelling interest 3
. Cancelling  #<Node 7>
. Cancelling  #<Node 6>
. Cancelling  #<Node 12>
. Cancelling  #<Node 5>
. Cancelling  #<Interest 4: ~(~p -> q) supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 6: ~(p -> q) supposing { q , ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 5: ~q supposing { ~p , ((p v q) & ((~p v q) & (p v ~q))) }>
. Cancelling  #<Interest 3: p supposing { ((p v q) & ((~p v q) & (p v ~q))) }>
.........................................................................................................................................

  # 12   ~q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~p }
  Inferred by support-link #9 from { 11 , 5 } by modus-ponens1
  undefeated-degree-of-support = 1.0

  # 13   p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #10 from { 6 , 12 } by reductio
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
16:    Retrieving #<Interest 7: q supposing { ((p v q) & ((~p v q) & (p v ~q))) }> from the inference-queue.
Preference = 1.0

  # 14   ~q    supposition: { ~q }
  supposition
---------------------------------------------------------------------------
17:    Retrieving #<Node 14> from the inference-queue.  Preference = 0.25
  Node 16 discharges interest 7
  Node 17 discharges interest 2
  Node 18 discharges interest 1

  # 15   ~p    supposition: { ((p v q) & ((~p v q) & (p v ~q))) , ~q }
  Inferred by support-link #11 from { 10 , 14 } by modus-tollens2
  undefeated-degree-of-support = 1.0

  # 16   q    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #12 from { 15 , 13 } by reductio
  undefeated-degree-of-support = 1.0

  # 17   ~(~p v ~q)    supposition: { ((p v q) & ((~p v q) & (p v ~q))) }
  Inferred by support-link #13 from { 13 , 16 } by i-neg-disj
  undefeated-degree-of-support = 1.0

  # 18   (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
  Inferred by support-link #14 from { 17 } by conditionalization
  undefeated-degree-of-support = 1.0

            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.


================= ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (((p v q) & ((~p v q) & (p v ~q))) -> ~(~p v ~q))
  is answered affirmatively by node 18
---------------------------------------------------

Cumulative size of arguments = 17
Size of inference-graph = 18 of which 0 were unused suppositions.
94% of the inference-graph was used in the argument.
7 interests were adopted.
4 suppositions were made.
```

3. Combining Deductive and Defeasible Reasoning

Given the ability to perform both deductive and defeasible reasoning, the next step is to combine both in a single problem. The natural suggestion is that we just apply the inference rules described in section two to problems that contain defeasible inferences as well as deductive inferences. But when we do that, surprising difficulties arise. For instance, consider the following simple problem, investigated without allowing OSCAR the use of *reductio*:

```
Problem #9
```

Figure 8 -- the lottery paradox paradox using logic

```
Given premises:
     P    justification = 1.0
Ultimate epistemic interests:
     ~T1    interest = 1.0
     ~T2    interest = 1.0
     ~T3    interest = 1.0

    FORWARDS PRIMA FACIE REASONS
      pf-reason_1.9:   {R} ||=> ~T1   strength = 1.0
      pf-reason_2.9:   {R} ||=> ~T2   strength = 1.0
      pf-reason_3.9:   {R} ||=> ~T3   strength = 1.0
      pf-reason_4.9:   {P} ||=> R   strength = 1.0

    FORWARDS CONCLUSIVE REASONS
      con-reason_1.9:   {R} ||=> (T1 v (T2 v T3))   strength = 1.0


  # 1   P
  given
                                      # 1
                                      interest: ~T3
                                      This is of ultimate interest
                                      # 2
                                      interest: ~T2
                                      This is of ultimate interest
                                      # 3
                                      interest: ~T1
                                      This is of ultimate interest
---------------------------------------------------------------------------
1:    Retrieving #<Node 1> from the inference-queue.  Preference = 1.0
                                      # 4
                                      interest: (P @ R)
                                      Of interest as defeater for support-link 2
                                      # 5
                                      interest: ~R
                                      Of interest as defeater for support-link 2

  # 2   R
  Inferred by support-link #2 from { 1 } by pf-reason_4.9
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
2:    Retrieving #<Node 2> from the inference-queue.  Preference = 1.0
                                      # 6
                                      interest: (R @ ~T3)
                                      Of interest as defeater for support-link 4
                                      # 7
                                      interest: T3
                                      Of interest as defeater for support-link 4
  Node 4 discharges interest 1
                                      # 8
                                      interest: (R @ ~T2)
                                      Of interest as defeater for support-link 5
                                      # 9
                                      interest: T2
                                      Of interest as defeater for support-link 5
  Node 5 discharges interest 2
                                      # 10
                                      interest: (R @ ~T1)
                                      Of interest as defeater for support-link 6
                                      # 11
                                      interest: T1
                                      Of interest as defeater for support-link 6
  Node 6 discharges interest 3

  # 3   (T1 v (T2 v T3))
  Inferred by support-link #3 from { 2 } by con-reason_1.9
```

```
 undefeated-degree-of-support = 1.0

 # 4   ~T3
 Inferred by support-link #4 from { 2 } by pf-reason_3.9
 undefeated-degree-of-support = 1.0

 # 5   ~T2
 Inferred by support-link #5 from { 2 } by pf-reason_2.9
 undefeated-degree-of-support = 1.0

 # 6   ~T1
 Inferred by support-link #6 from { 2 } by pf-reason_1.9
 undefeated-degree-of-support = 1.0
        ========================================
        Justified belief in ~T3
        with undefeated-degree-of-support 1.0
        answers #<Query 3: ~T3>
        ========================================
        ========================================
        Justified belief in ~T2
        with undefeated-degree-of-support 1.0
        answers #<Query 2: ~T2>
        ========================================
        ========================================
        Justified belief in ~T1
        with undefeated-degree-of-support 1.0
        answers #<Query 1: ~T1>
        ========================================
--------------------------------------------------------------------------
3:    Retrieving #<Node 4> from the inference-queue.  Preference = 0.5
--------------------------------------------------------------------------
4:    Retrieving #<Node 5> from the inference-queue.  Preference = 0.5
--------------------------------------------------------------------------
5:    Retrieving #<Node 6> from the inference-queue.  Preference = 0.5
--------------------------------------------------------------------------
6:    Retrieving #<Node 3> from the inference-queue.  Preference = 0.2

 # 7   (~T1 -> (T2 v T3))
 Inferred by support-link #7 from { 3 } by disj-simp
 undefeated-degree-of-support = 1.0
--------------------------------------------------------------------------
7:    Retrieving #<Node 7> from the inference-queue.  Preference = 0.1666

 # 8   (T2 v T3)
 Inferred by support-link #8 from { 7 , 6 } by modus-ponens1
 undefeated-degree-of-support = 1.0
--------------------------------------------------------------------------
8:    Retrieving #<Node 8> from the inference-queue.  Preference = 0.3333

 # 9   (~T2 -> T3)
 Inferred by support-link #9 from { 8 } by disj-simp
 undefeated-degree-of-support = 1.0
--------------------------------------------------------------------------
9:    Retrieving #<Node 9> from the inference-queue.  Preference = 0.25

 # 10   T3              DEFEATED
 Inferred by support-link #10 from { 9 , 5 } by modus-ponens1
 defeatees: { link 4 for node 4 }
 undefeated-degree-of-support = 0.0

 # 11   T2              DEFEATED
 Inferred by support-link #11 from { 2 , 6 , 4 } by inversion_from_contradictory_nodes_10_and_4
 defeatees: { link 5 for node 5 }
 undefeated-degree-of-support = 0.0

 # 12   T1              DEFEATED
 Inferred by support-link #12 from { 2 , 4 , 5 } by inversion_from_contradictory_nodes_11_and_5
 defeatees: { link 6 for node 6 }
 undefeated-degree-of-support = 0.0

 # 13   ~R                  DEFEATED
 Inferred by support-link #13 from { 4 , 5 , 6 } by inversion_from_contradictory_nodes_12_and_6
 defeatees: { link 2 for node 2 }
 undefeated-degree-of-support = 0.0
---------------------------------------------------------------------------
Affected-nodes:
(#<Node 7> #<Node 3> #<Node 2> #<Node 9> #<Node 8> #<Node 6> #<Node 10> #<Node 5>
#<Node 13> #<Node 12> #<Node 11> #<Node 4>)
---------------------------------------------------------------------------
creating:  #<triangle-set 1>  #<assignment-tree 4>  #<assignment-tree 3>  #<assignment-tree 2>
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 1:
. ((#<Node 1> . 1.0))
. triangle 1:
. . (#<support-link #2 for node 2> #<support-link #3 for node 3> #<support-link #7 for node 7>
#<support-link #8 for node 8> #<support-link #9 for node 9>
. .   #<support-link #10 for node 10> #<support-link #11 for node 11> #<support-link #12 for node 12>
#<support-link #6 for node 6> #<support-link #5 for node 5>
. .   #<support-link #4 for node 4> #<support-link #13 for node 13>)
. . . . assignment-tree 2:
. . . . . ((#<Node 1> . 1.0) (#<Node 13> . 0.0) (#<Node 12> . 0.0) (#<Node 11> . 0.0)
. . . . .    (#<Node 4> . 0.0) (#<Node 10> . 1.0) (#<Node 5> . 1.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 9> . 1.0) (#<Node 8> . 1.0) (#<Node 6> . 1.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 7> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0))
. . . . assignment-tree 3:
. . . . . ((#<Node 1> . 1.0) (#<Node 11> . 1.0) (#<Node 4> . 1.0) (#<Node 13> . 0.0)
. . . . .    (#<Node 12> . 0.0) (#<Node 10> . 0.0) (#<Node 5> . 0.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 9> . 1.0) (#<Node 8> . 1.0) (#<Node 6> . 1.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 7> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0))
. . . . assignment-tree 4:
. . . . . ((#<Node 1> . 1.0) (#<Node 12> . 1.0) (#<Node 4> . 1.0) (#<Node 10> . 0.0)
. . . . .    (#<Node 5> . 1.0) (#<Node 9> . 0.0) (#<Node 13> . 0.0) (#<Node 11> . 0.0)
. . . . .    (#<Node 8> . 0.0) (#<Node 6> . 0.0) (#<Node 1> . 1.0) (#<Node 7> . 1.0)
. . . . .    (#<Node 3> . 1.0) (#<Node 2> . 1.0))
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 10> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 11> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 12> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 13> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 4> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 5> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 9> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 8> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 6> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              ===========================================
              Lowering the undefeated-degree-of-support of ~T3
              retracts the previous answer to #<Query 3: ~T3>
              ===========================================
              ===========================================
              Lowering the undefeated-degree-of-support of ~T2
              retracts the previous answer to #<Query 2: ~T2>
              ===========================================
              ===========================================
              Lowering the undefeated-degree-of-support of ~T1
```

```
               retracts the previous answer to #<Query 1: ~T1>
               =========================================
---------------------------------------------------------------------------
10:    Retrieving #<Node 10> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
11:    Retrieving #<Node 11> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
12:    Retrieving #<Node 12> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
13:    Retrieving #<Interest 11: T1> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
14:    Retrieving #<Interest 9: T2> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
15:    Retrieving #<Interest 7: T3> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
16:    Retrieving #<Node 13> from the inference-queue.  Preference = 0.05
---------------------------------------------------------------------------
17:    Retrieving #<Interest 5: ~R> from the inference-queue.  Preference = 0.05


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in ~T3
  is unsatisfied.
---------------------------------------------------
  Interest in ~T2
  is unsatisfied.
---------------------------------------------------
  Interest in ~T1
  is unsatisfied.
---------------------------------------------------

Cumulative size of arguments = 13
Size of inference-graph = 13 of which 0 were unused suppositions.
100% of the inference-graph was used in the argument.
11 interests were adopted.
0 suppositions were made.
```

If we allow OSCAR the use of *reductio,* we get the following:

```
Problem #9
Figure 8 -- the lottery paradox paradox using logic
Given premises:
     P    justification = 1.0
Ultimate epistemic interests:
     ~T1    interest = 1.0
     ~T2    interest = 1.0
     ~T3    interest = 1.0

    FORWARDS PRIMA FACIE REASONS
      pf-reason_1.9:   {R} ||=> ~T1   strength = 1.0
      pf-reason_2.9:   {R} ||=> ~T2   strength = 1.0
      pf-reason_3.9:   {R} ||=> ~T3   strength = 1.0
      pf-reason_4.9:   {P} ||=> R   strength = 1.0

    FORWARDS CONCLUSIVE REASONS
      con-reason_1.9:   {R} ||=> (T1 v (T2 v T3))   strength = 1.0


  # 1   P
  given
                                       # 1
                                       interest: ~T3
                                       This is of ultimate interest

  # 2   T3    supposition: { T3 }
  reductio-supposition
```

```
                                 Readopting interest in:
                                      # 1
                                      reductio interest: ~T3
                                      This is of ultimate interest
                                      # 2
                                      interest: ~T2
                                      This is of ultimate interest

  # 3   T2    supposition: { T2 }
  reductio-supposition
                                 Readopting interest in:
                                      # 2
                                      reductio interest: ~T2
                                      This is of ultimate interest
                                      # 3
                                      interest: ~T1
                                      This is of ultimate interest

  # 4   T1    supposition: { T1 }
  reductio-supposition
                                 Readopting interest in:
                                      # 3
                                      reductio interest: ~T1
                                      This is of ultimate interest
---------------------------------------------------------------------------
1:    Retrieving #<Node 1> from the inference-queue.  Preference = 1.0
                                      # 4
                                      interest: (P @ R)
                                      Of interest as defeater for support-link 2
                                      # 5
                                      interest: ~R
                                      Of interest as defeater for support-link 2
                                      # 6
                                      reductio interest: ~P
                                      using node 1

  # 5   R
  Inferred by support-link #2 from { 1 } by pf-reason_4.9
  undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
2:    Retrieving #<Node 5> from the inference-queue.  Preference = 1.0
                                      # 7
                                      interest: (R @ ~T3)
                                      Of interest as defeater for support-link 4
                                      # 8
                                      interest: T3
                                      Of interest as defeater for support-link 4
  Node 7 discharges interest 1
                                      # 9
                                      interest: (R @ ~T2)
                                      Of interest as defeater for support-link 5
                                      # 10
                                      interest: T2
                                      Of interest as defeater for support-link 5
  Node 8 discharges interest 2
                                      # 11
                                      interest: (R @ ~T1)
                                      Of interest as defeater for support-link 6
                                      # 12
                                      interest: T1
                                      Of interest as defeater for support-link 6
  Node 9 discharges interest 3
                                 Readopting interest in:
                                      # 5
                                      reductio interest: ~R

  # 6   (T1 v (T2 v T3))
```

```
 Inferred by support-link #3 from { 5 } by con-reason_1.9
 undefeated-degree-of-support = 1.0

 # 7   ~T3
 Inferred by support-link #4 from { 5 } by pf-reason_3.9
 undefeated-degree-of-support = 1.0

 # 8   ~T2
 Inferred by support-link #5 from { 5 } by pf-reason_2.9
 undefeated-degree-of-support = 1.0

 # 9   ~T1
 Inferred by support-link #6 from { 5 } by pf-reason_1.9
 undefeated-degree-of-support = 1.0
            =========================================
            Justified belief in ~T3
            with undefeated-degree-of-support 1.0
            answers #<Query 3: ~T3>
            =========================================
            =========================================
            Justified belief in ~T2
            with undefeated-degree-of-support 1.0
            answers #<Query 2: ~T2>
            =========================================
            =========================================
            Justified belief in ~T1
            with undefeated-degree-of-support 1.0
            answers #<Query 1: ~T1>
            =========================================
---------------------------------------------------------------------------
3:    Retrieving #<Node 7> from the inference-queue.  Preference = 0.5
                                Readopting interest in:
                                    # 8
                                    reductio interest: T3
---------------------------------------------------------------------------
4:    Retrieving #<Node 8> from the inference-queue.  Preference = 0.5
                                Readopting interest in:
                                    # 10
                                    reductio interest: T2
---------------------------------------------------------------------------
5:    Retrieving #<Node 9> from the inference-queue.  Preference = 0.5
                                Readopting interest in:
                                    # 12
                                    reductio interest: T1
---------------------------------------------------------------------------
6:    Retrieving #<Interest 6: ~P> from the inference-queue.  Preference = 0.5
---------------------------------------------------------------------------
7:    Retrieving #<Node 4> from the inference-queue.  Preference = 0.5
 Node 4 discharges interest 12
---------------------------------------------------------------------------
8:    Retrieving #<Node 3> from the inference-queue.  Preference = 0.5
 Node 3 discharges interest 10
---------------------------------------------------------------------------
9:    Retrieving #<Node 2> from the inference-queue.  Preference = 0.5
 Node 2 discharges interest 8
---------------------------------------------------------------------------
10:    Retrieving #<Node 6> from the inference-queue.  Preference = 0.2

 # 10   (~T1 -> (T2 v T3))
 Inferred by support-link #7 from { 6 } by disj-simp
 undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
11:    Retrieving #<Node 10> from the inference-queue.  Preference = 0.1666
                                    # 13
                                    reductio interest: ~(~T1 -> (T2 v T3))
                                    using node 10
 # 11   (T2 v T3)
 Inferred by support-link #8 from { 10 , 9 } by modus-ponens1
 undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
12:    Retrieving #<Node 11> from the inference-queue.  Preference = 0.3333

 # 12   (~T2 -> T3)
 Inferred by support-link #9 from { 11 } by disj-simp
 undefeated-degree-of-support = 1.0
---------------------------------------------------------------------------
13:    Retrieving #<Node 12> from the inference-queue.  Preference = 0.25
 Node 13 discharges interest 8
 Node 14 discharges interest 10
 Node 15 discharges interest 12
 Node 16 discharges interest 5
                                   # 14
                                   reductio interest: ~(~T2 -> T3)
                                   using node 12

 # 13   T3                DEFEATED
 Inferred by support-link #10 from { 12 , 8 } by modus-ponens1
 defeatees: { link 4 for node 7 }
 undefeated-degree-of-support = 0.0

 # 14   T2                DEFEATED
 Inferred by support-link #11 from { 5 , 9 , 7 } by inversion_from_contradictory_nodes_13_and_7
 defeatees: { link 5 for node 8 }
 undefeated-degree-of-support = 0.0

 # 15   T1                DEFEATED
 Inferred by support-link #12 from { 5 , 7 , 8 } by inversion_from_contradictory_nodes_14_and_8
 defeatees: { link 6 for node 9 }
 undefeated-degree-of-support = 0.0

 # 16   ~R                DEFEATED
 Inferred by support-link #13 from { 7 , 8 , 9 } by inversion_from_contradictory_nodes_15_and_9
 defeatees: { link 2 for node 5 }
 undefeated-degree-of-support = 0.0
---------------------------------------------------------------------------
Affected-nodes:
(#<Node 10> #<Node 6> #<Node 5> #<Node 12> #<Node 11> #<Node 9> #<Node 13> #<Node 8>
#<Node 16> #<Node 15> #<Node 14> #<Node 7>)
---------------------------------------------------------------------------
creating:  #<triangle-set 1>  #<assignment-tree 4>  #<assignment-tree 3>  #<assignment-tree 2>
---------------------------------------------------------------------------
Recomputed assignment-tree:
assignment-tree 1:
. ((#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0) (#<Node 1> . 1.0))
. triangle 1:
. . (#<support-link #2 for node 5> #<support-link #3 for node 6> #<support-link #7 for node 10>
#<support-link #8 for node 11> #<support-link #9 for node 12>
. .   #<support-link #10 for node 13> #<support-link #11 for node 14> #<support-link #12 for node 15>
#<support-link #6 for node 9> #<support-link #5 for node 8>
. .   #<support-link #4 for node 7> #<support-link #13 for node 16>)
. . . . assignment-tree 2:
. . . . . ((#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 16> . 0.0) (#<Node 15> . 0.0) (#<Node 14> . 0.0) (#<Node 7> . 0.0)
. . . . .    (#<Node 13> . 1.0) (#<Node 8> . 1.0) (#<Node 4> . 1.0) (#<Node 3> . 1.0)
. . . . .    (#<Node 2> . 1.0) (#<Node 1> . 1.0) (#<Node 12> . 1.0) (#<Node 11> . 1.0)
. . . . .    (#<Node 9> . 1.0) (#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0)
. . . . .    (#<Node 1> . 1.0) (#<Node 10> . 1.0) (#<Node 6> . 1.0) (#<Node 5> . 1.0))
. . . . assignment-tree 3:
. . . . . ((#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0) (#<Node 1> . 1.0)
. . . . .    (#<Node 14> . 1.0) (#<Node 7> . 1.0) (#<Node 16> . 0.0) (#<Node 15> . 0.0)
. . . . .    (#<Node 13> . 0.0) (#<Node 8> . 0.0) (#<Node 4> . 1.0) (#<Node 3> . 1.0)
. . . . .    (#<Node 2> . 1.0) (#<Node 1> . 1.0) (#<Node 12> . 1.0) (#<Node 11> . 1.0)
. . . . .    (#<Node 9> . 1.0) (#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0)
. . . . .   (#<Node 1> . 1.0) (#<Node 10> . 1.0) (#<Node 6> . 1.0) (#<Node 5> . 1.0))
. . . . assignment-tree 4:
. . . . . ((#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0) (#<Node 1> . 1.0)
. . . . .   (#<Node 15> . 1.0) (#<Node 7> . 1.0) (#<Node 13> . 0.0) (#<Node 8> . 1.0)
. . . . .   (#<Node 12> . 0.0) (#<Node 16> . 0.0) (#<Node 14> . 0.0) (#<Node 11> . 0.0)
. . . . .   (#<Node 9> . 0.0) (#<Node 4> . 1.0) (#<Node 3> . 1.0) (#<Node 2> . 1.0)
. . . . .   (#<Node 1> . 1.0) (#<Node 10> . 1.0) (#<Node 6> . 1.0) (#<Node 5> . 1.0))
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 13> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 14> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 15> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 16> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 7> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 8> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 12> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 11> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 9> has become defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              =========================================
              Lowering the undefeated-degree-of-support of ~T3
              retracts the previous answer to #<Query 3: ~T3>
              =========================================
              =========================================
              Lowering the undefeated-degree-of-support of ~T2
              retracts the previous answer to #<Query 2: ~T2>
              =========================================
              =========================================
              Lowering the undefeated-degree-of-support of ~T1
              retracts the previous answer to #<Query 1: ~T1>
              =========================================
---------------------------------------------------------------------------
14:    Retrieving #<Interest 14: ~(~T2 -> T3)> from the inference-queue.  Preference = 0.2
                              Readopting interest in:
                                   # 1
                                   reductio interest: ~T3
                                   This is of ultimate interest
                                   For reductio interest 14 by i-neg-condit
                                   Node #7 discharges interest #1
                              Readopting interest in:
                                   # 2
                                   reductio interest: ~T2
                                   This is of ultimate interest
                                   For reductio interest 14 by i-neg-condit using node 7
                                   Node #8 discharges interest #2
  Node 17 discharges interest 14

  # 17   ~(~T2 -> T3)                 DEFEATED
  Inferred by support-link #14 from { 7 , 8 } by i-neg-condit
  undefeated-degree-of-support = 0.0
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              #<Node 17> is defeated.
              vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
---------------------------------------------------------------------------
15:    Retrieving #<Interest 13: ~(~T1 -> (T2 v T3))> from the inference-queue.  Preference = 0.1428
                                   # 15
                                   reductio interest: ~(T2 v T3)
                                   For reductio interest 13 by i-neg-condit
---------------------------------------------------------------------------
16:    Retrieving #<Interest 15: ~(T2 v T3)> from the inference-queue.  Preference = 0.25
```

```
                                 Readopting interest in:
                                      # 2
                                      reductio interest: ~T2
                                      This is of ultimate interest
                                      For reductio interest 15 by i-neg-disj
                                      For reductio interest 14 by i-neg-condit using node 7
                                      Node #8 discharges interest #2
                                 Readopting interest in:
                                      # 1
                                      reductio interest: ~T3
                                      This is of ultimate interest
                                      For reductio interest 15 by i-neg-disj using node 8
                                      For reductio interest 14 by i-neg-condit
                                      Node #7 discharges interest #1
 Node 18 discharges interest 15
                                 Readopting interest in:
                                      # 3
                                      reductio interest: ~T1
                                      This is of ultimate interest
                                      For reductio interest 13 by i-neg-condit using node 18
                                      Node #9 discharges interest #3
 Node 19 discharges interest 13

 # 18   ~(T2 v T3)                  DEFEATED
 Inferred by support-link #15 from { 8 , 7 } by i-neg-disj
 undefeated-degree-of-support = 0.0

 # 19   ~(~T1 -> (T2 v T3))                  DEFEATED
 Inferred by support-link #16 from { 18 , 9 } by i-neg-condit
 undefeated-degree-of-support = 0.0
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            #<Node 18> is defeated.
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            #<Node 19> is defeated.
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
---------------------------------------------------------------------------
17:    Retrieving #<Node 13> from the inference-queue.  Preference = 0.1
                                 Readopting interest in:
                                      # 1
                                      reductio interest: ~T3
                                      This is of ultimate interest
                                      For reductio interest 15 by i-neg-disj using node 8
                                      For reductio interest 14 by i-neg-condit
---------------------------------------------------------------------------
18:    Retrieving #<Node 14> from the inference-queue.  Preference = 0.1
                                 Readopting interest in:
                                      # 2
                                      reductio interest: ~T2
                                      This is of ultimate interest
                                      For reductio interest 15 by i-neg-disj
                                      For reductio interest 14 by i-neg-condit using node 7
---------------------------------------------------------------------------
19:    Retrieving #<Node 15> from the inference-queue.  Preference = 0.1
                                 Readopting interest in:
                                      # 3
                                      reductio interest: ~T1
                                      This is of ultimate interest
                                      For reductio interest 13 by i-neg-condit using node 18
---------------------------------------------------------------------------
20:    Retrieving #<Interest 12: T1> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
21:    Retrieving #<Interest 10: T2> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
22:    Retrieving #<Interest 8: T3> from the inference-queue.  Preference = 0.1
---------------------------------------------------------------------------
23:    Retrieving #<Node 16> from the inference-queue.  Preference = 0.05
                                      # 16
                                 reductio interest: R
                                 using node 16
---------------------------------------------------------------------------
24:    Retrieving #<Interest 16: R> from the inference-queue.  Preference = 1.0
---------------------------------------------------------------------------
25:    Retrieving #<Interest 5: ~R> from the inference-queue.  Preference = 0.05
---------------------------------------------------------------------------
26:    Retrieving #<Node 18> from the inference-queue.  Preference = 0.025
---------------------------------------------------------------------------
27:    Retrieving #<Node 17> from the inference-queue.  Preference = 0.02
---------------------------------------------------------------------------
28:    Retrieving #<Node 19> from the inference-queue.  Preference = 0.0142

Cumulative size of arguments = 13
Size of inference-graph = 19 of which 0 were unused suppositions.
73% of the inference-graph was used in the argument.
16 interests were adopted.
3 suppositions were made.
```

Exactly the same arguments are produced, but OSCAR draws 50% more conclusions and adopts 50% more interests. In addition, most of the interests adopted for non-reductio reasoning are readopted as reductio-interests. The reasoning takes 66% longer. Some of this reasoning is essential. There is no way to know beforehand whether interests might be provable deductively, so reductio-suppositions 2, 3, and 4 cannot be avoided, nor can the conversion of interests 1, 2, and 3 into reductio-interests. But all of the rest of the extra reasoning (printed in boldface) is pointless. For example, when R is inferred defeasibly at node 5, a reductio interest in ~R is adopted. (A non-reductio interest in ~R is already present as an interest in a rebutting defeater for node 5.) But even if we could infer ~R deductively, that would not enable us to make a reductio inference. All it would do would be to defeat the inference to R. We could not then use R and ~R jointly to infer something else, because any such argument would be self-defeating. Similarly, having inferred (~T2 -> T3) defeasibly at node 12, OSCAR adopts interest in ~(~T2 -> T3) and succeeds in getting it at node 17, but that is pointless because even given the contradiction, no reductio inference can be drawn.

This suggests imposing constraints on the use of *reductio* that have the effect of restricting it to deductive reasoning. Four such constraints can be imposed. First, we add a new slot to inference-nodes:

- deductive-only — if this is T, only non-defeasible reasons can be used in making inferences from this node.

Reductio-suppositions are marked deductive-only.

Second, we add a slot to interests:

- non-reductio-interest — T if the interest has a non-reductio source.

Note that an interest can be both a reductio-interest and a non-reductio-interest if it is generated in more than one way. The second constraint is then that only deductive-conclusions can discharge an interest that is not non-reductio.

Third, only deductive nodes generate reductio-interests.

The fourth constraint may be controversial. In many problems, premises or percepts play the role of background knowledge, supplying information to be used by other reasoning problems. It seems that reductio-reasoning should not lead to reductio-interests in the negation of background knowledge. To implement this, we mark whatever we want as background-knowledge (using the slot provided in inference-nodes), and require that any node counts as background knowledge if it has a support-link in which all members of the support-link basis are background knowledge. We then redefine deductive-nodes do be those having NIL as a nearest-defeasible-ancestor (i.e., nodes supported by a deductive argument) *and not being background knowledge*. When combined with the previous constraints, this has the effect of precluding background knowledge from reductio reasoning.

With the adoption of these constraints, none of the boldfaced reasoning in the previous example is performed. These constraints seem to be adequate to make *reductio* well behaved in the context of mixed deductive/defeasible reasoning.