V
FIRST-ORDER REASONING

1. Simple Rules for the Predicate Calculus

Turning to first-order logic, reasoning with quantifiers can be handled in a number of different ways. The simplest strategy is to adopt rules introducing and eliminating negations of quantified formulas and rules for existential and universal instantiation and generalization. Universal generalization (UG) and existential generalization (EG) are interest rules. UG instructs the reasoner that if it is interested in a universal generalization $(\forall x)(F\ x)$, it is to manufacture a new free variable $\hat{}x$ and adopt interest in $F\hat{}x$. EG instructs the reasoner that if it is interested in an existential generalization $(\exists x)(F\ x)$, it should adopt interest in instances $(F\ c)$ of the scope of the quantifier. UI and EI are forwards rules. UI licenses inferences from universal generalizations $(\forall x)(F\ x)$ to instances $(F\ c)$. EI instructs the reasoner to manufacture a new free variable *@x* and infer *F@x* from $(\exists x)(F\ x)$. There are various options regarding how, precisely, to formulate UI and EG. These concern what instances to infer from a universal generalization, and in what instances of an existential generalization to adopt interest.

Consider UI. If the reasoner concludes $(\forall x)(F\ x)/X$, it cannot immediately infer every instance $(F\ c)/X$ from this, because there are potentially infinitely many instances (this is automatically true when reasoning in a context like mathematics in which there are function symbols). A simple solution to this problem (adopted in Pollock [1990a]) is to infer every instance $(F\ c)/X$ where c is a term already occurring in some conclusion P/Y such that $Y \subseteq X$. This must be augmented with an auxiliary rule instructing the reasoner that when $(\forall x)(F\ x)/X$ has been concluded and then a new conclusion P/Y is drawn such that $Y \subseteq X$, and c occurs in P but has not previously occurred in conclusions drawn relative to suppositions that are contained in X, then the conclusion $(F\ c)/X$ should be drawn. Notice that this is an auxiliary rule, and not an inference rule, because $(F\ c)$ is not being inferred from P. Instead, the inference of $(F\ c)$ from $(\forall x)(F\ x)$ is "triggered" by concluding P. This is, perhaps, an undesirable feature of first-order logic, where terms automatically have denotations. In a free logic, where '*c*' could fail to have a denotation, this rule would require that P imply $(E!\ c)$ ("*c exists*"), and in that case, P would provide part of the basis for the inference and this would be an ordinary inference rule.

This auxiliary rule represents another place in which discount-factors must be employed. Suppose, for example, that the reasoner concludes $(\forall x)(\exists y)(F\ x\ y)$, and '*c*' has already occurred. By UI, the reasoner will conclude $(\exists y)(F\ c\ y)$, then by EI it will conclude $(F\ c\ @y)$; then by UI $(\exists y)(F\ @y\ y)$, and by EI $(F\ @y\ @@y)$, and so on without limit. To prevent this repetitive reasoning from monopolizing system resources and thereby interfering with other reasoning, its priority on the *inference-queue* must be lowered by an appropriate discount-factor.

Another (possibly undesirable) feature of first-order logic is that the domain over which the variables range is required to be nonempty. This is a second respect in which first-order logic contrasts with free logic. This has the consequence that if the quantifier rules are to generate all of the theorems of first-order logic, there must always be at least one term for UI to use for instantiations. This can be accomplished by introducing a dummy term '@' when no other terms have occurred.

OSCAR can thus perform reasoning in standard first-order logic by adding the following rules to those for the propositional calculus (where Sb(*c*,*x*)*P* is the result of substituting '*c*' for all free occurrences of '*x*' in *P*):

Forwards reasons:

quantifier negation eliminations:

infer $(\exists x)\neg P$ from $\sim(\forall x)P$

infer $(\forall x)\neg P$ from $\sim(\exists x)P$

universal instantiation:

infer $\mathrm{Sb}(c,x)P/X$ from $(\forall x)P/X$ where c is a term already occurring in some conclusion Q/Y such that $Y \subseteq X$ and $\mathrm{Sb}(c,x)P$ results from substituting c for all free occurrences of x in P. If there are no such terms, infer $\mathrm{Sb}(@,x)P/X$ from $(\forall x)P/X$.

existential instantiation:

infer $\mathrm{Sb}(@x,x)P/X$ from $(\exists x)P/X$ where $@x$ is a constant that has not previously occurred in any conclusions.

Auxiliary rule for forwards reasoning:

If Q/Y is a newly adopted conclusion, then for each conclusion of the form $(\forall x)P/X$ such that $Y \subseteq X$, infer $\mathrm{Sb}(c,x)P/X$ from $(\forall x)P/X$ where c is a term occurring in Q/Y but not occurring in any previous conclusions.

Backwards reasons:

quantifier negation introductions:

adopt interest in $(\exists x)\neg P$ to infer $\sim(\forall x)P$

adopt interest in $(\forall x)\neg P$ to infer $\sim(\exists x)P$

universal generalization:

adopt interest in $\mathrm{Sb}(\hat{}x,x)P/X$ to infer $(\forall x)P/X$, where $\hat{}x$ is a free variable that has not previously occurred in any conclusions.

existential generalization:

adopt interest in $\mathrm{Sb}(c,x)P/X$ to infer $(\exists x)P/X$ where c is a term already occurring in some conclusion Q/Y such that $Y \subseteq X$. If there are no such terms, adopt interest in $\mathrm{Sb}(@,x)P/X$ to infer $(\exists x)P/X$.

Auxiliary rule for backwards reasoning:

If Q/Y is a newly adopted conclusion, then for each interest of the form $(\exists x)P/X$ such that $Y \subseteq X$, adopt interest in $\mathrm{Sb}(c,x)P/X$ to infer $(\exists x)P/X$ where c is a term occurring in Q/Y but not occurring in any previous conclusions.

In addition, the syntactical constraint on *reductio* must be liberalized to allow its use with quantified formulas.

These rules make OSCAR complete for first-order logic. That is, given any valid formula P of first-order logic, if it is made the query-formula of a query inserted into *ultimate-epistemic-interests*, OSCAR will construct a deductive argument supporting P and thereby come to believe P. These rules are not, however, terribly efficient. In my [1990a] I discussed a rule called *all-detachment* that is a more efficient form of universal instantiation and improves the efficiency of reasoning in first-order logic without function symbols. For reasoning in first-order logic with function symbols, it seems reasonably clear that some more "intelligent" form of instantiation rules is required. UI and EG proceed by systematically trying all instances of the quantified formulas, but once we have function symbols at our disposal, there are too many instances for this to be feasible. Most automated theorem provers use some form of unification to try to compute which instances may be useful.[1] In principle, any of these techniques can be incorporated into OSCAR by giving OSCAR the right inference rules.

The above rules for first-order reasoning enable OSCAR to answer definite queries formulated in the predicate calculus. But OSCAR must also be able to answer indefinite

[1] For surveys of this material, see Bundy [1983] and Wos, et al [1984].

queries — queries of the form ⌜Find an x such that $(F\ x)$⌝. For this purpose, I will introduce a new quantifier ⌜$(?x)$⌝, and symbolize this query-formula as ⌜$(?x)(F\ x)$⌝. Attempts to answer this query can then be handled by adding the following two rules to OSCAR's first-order inference rules:

Backwards reason:

indefinite query answering:

adopt interest in Sb(c,x)P/X to infer $(?x)P/X$ where c is a term already occurring in some conclusion Q/Y such that $Y \subseteq X$.

Auxiliary rule for backwards reasoning:

If Q/Y is a newly adopted conclusion, then for each interest of the form $(?x)P/X$ such that $Y \subseteq X$, adopt interest in Sb(c,x)P/X to infer $(?x)P/X$ where c is a term occurring in Q/Y but not occurring in any previous conclusions.

Just as for ordinary first-order reasoning, these rules are "theoretically adequate", but too inefficient for many purposes. For example, if we employ these rules in a mathematical context in which we are looking for a construction that satisfies certain conditions, the reasoner will proceed blindly by simply trying every term it encounters. That will only work in very simple cases. Realistically, we need more intelligent instantiation rules that will focus the reasoner's attention on "plausible candidates" for constructions satisfying the given conditions.

2. Unification

The preceding instantiation rules have the effect that basically the same reasoning will be repeated over and over again. For instance, given an interest in $(\exists x)(F\ x)$, the reasoner will adopt interest in $(F\ a)$, and reason backwards from it. Then the reasoner will also adopt interest in $(F\ b)$, and perform essentially the same backwards reasoning from that formula. If we assume that reason-schemas just concern predicates and relations, and do not contain occurrences of individual constants, then nothing would be lost by keeping the reasoning general, adopting an interest in $(F\ x)$, for free x, rather than separate interests in $(F\ a)$ and $(F\ b)$, reason backwards from $(F\ x)$, and look for bindings for x that would satisfy the resulting interests.

It will occur to those familiar with the current literature in automated reasoning that this problem can be solved by using skolemization and unification. Skolemization and unification are well understood for deductive systems based upon resolution refutation. However, there are novel problems for how to use them in a natural deduction system, and as will become apparent below, these problems are not trivial.

Unification in forwards-reasoning

The basic strategy for using unification in forwards-reasoning is to have the rules of UI and EI replace bound variables by free variables and skolem functions (explained below), and then use unification to find common instances of formulas containing free variables.

Two expressions E1 and E2 containing variables *unify* iff there is a set of assignments to the variables of each which produces a common expression. For example, where x and y are variables, $(F\ x\ a)$ and $(F\ b\ y)$ unify to $(F\ b\ a)$ by the assignment $x = b$ and $y = a$. There may be more than one way to unify a pair of expressions. For instance, $(F\ x)$ and $(F\ y)$ can be unified either by assigning y to x (producing $(F\ x)$) or by assigning x to y (producing $(F\ y)$). The latter expressions could also be unified by assigning a to both x and y, producing $(F\ a)$. The latter unification is "less general" than either of the former, in the sense that $(F\ a)$ is an instance of either $(F\ x)$ or $(F\ y)$, but not conversely. A *most general unifier* is one that produces an expression such that the expression produced by any other unifier is an instance of it. In this example, the unifier that assigns x to y and the unifier that assigns y to x are both most general unifiers.

In the limiting case in which two expressions are identical, their unifier will be taken to be T. Otherwise, unifiers will be taken to be a-lists (lists of dotted pairs), and as such are applied to expressions by substituting the second member of each dotted-pair for the first member. The proper way to write this in LISP is (SUBLIS *u expression*), however I will often abbreviate this as *u*(*expression*).

If two expressions contain disjoint sets of variables, then a unifier can be regarded as a single assignment to all of the variables simultaneously. In this case, there are standard algorithms for computing a most general unifier if one exists. As a first approximation, we can let (MGU *p q vars*) be a most general unifier for *p* and *q* relative to the variables in *vars* (where *p* and *q* have no variables in common).

In the standard definition of unification, the parts of the expressions that are not variables must be identical in order for the expressions to unify. However, when dealing with quantified formulas, it is useful to allow them to be notational variants rather than requiring them to be identical. Then, for example, we can have $(\exists y)(F\ a\ y)$ unify with $(\exists z)(F\ x\ z)$ even though the bound variables *y* and *z* are different. It is straightforward to build a test for notational variants into the unification algorithms, and I will so-understand MGU.

Expressions can only be unified by MGU if they contain no variables in common. If they do share variables, those common variables must first be rewritten, and the unifier must be regarded as a pair (u_1,u_2) of assignments rather than a single assignment (thus allowing the same variable to be treated in two different ways in the two expressions). (UNIFIER *p q p-vars q-vars*) produces this pair of assignments when *p-vars* are the variables occurring in *p* and *q-vars* are the variables occurring in *q*.

We will also have occasion to unify one *set* of expressions *into* another. This amounts to finding all unifiers that unify the first set with some subset of the second set. This is accomplished by the function SET-UNIFIER. (SET-UNIFIER *X Y X-vars Y-vars*) returns the list of all unifiers unifying *X* into *Y*. For example, (set-unifier '((F c) (G y)) '((G a) (H c) (G b) (F z)) '(x y) '(z)) returns ((((y . a)) ((z . c))) (((y . b)) ((z . c)))).

Formulas with free variables will be produced by the forwards-inference-rules UI and EI. These rules instantiate quantified conclusions by replacing universally quantified variables by free variables, and existentially quantified variables by skolem-constants (if the formula contains no free variables) or skolem-functions applied to the free variables of the formula. Typographically, skolem-constants and skolem-functions will be indicated by appending "@". For instance, the terms "@*y*" and "(@*y x z*)" might be produced by EI. A record of the resulting free variables occurring in the node-formula is kept in the new slot *node-variables* . For convenience, we also add a slot *c-list-variables* to c-lists and record the list of node-variables there as well. EI and UI are then formulated as follows (sections of pseudo-code printed against a grey background will be discussed later in the chapter):

```
UI node
    If node is of type :inference, and its node-formula p is a universal generalization, let x
    be the variable bound by the initial universal quantifier and let p* be the matrix of p
    (the formula immediately following the initial quantifier). Construct a new variable y
    and substitute it for x in p* to produce p**. Then DRAW-CONCLUSION p** {node} UI ⟨T⟩
    1.0.

EI node
    If node is of type :inference, and its node-formula p is an existential generalization:
    • Let x be the variable bound by the initial existential quantifier, let p* be the
      matrix of p (the formula immediately following the initial quantifier), let u-vars
      be the list of node-variables of node, and let s-funs be the list of skolem-functions
      occurring in p*.
        • If s-funs is empty, let level be 1. Otherwise, let level be one plus the maximum
          of the ei-levels of the members of s-funs. Let discount be $.5^{(level-1)}$.
        • If u-vars is empty, let fun be a new variable @y. Let term be fun  If u-vars is
          nonempty, let fun be a new skolem-function @y, and let term be be construct-
          ed by appending @y to the front of the list of u-vars.
        • Let p** be the result of substituting term for x in p*.
        • If there is no node node* in processed-conclusions whose node-formula differs
```

from p^{**} only by having some other singular term in place of *term*:

- set the ei-level of *fun* to be *level*
- DRAW-CONCLUSION p^{**} {*node*} EI ⟨T⟩ *discount*.

The free variables produced by UI will be called *conclusion-variables*, to distinguish them from the free-variables (*interest-variables*) that will be produced (see below) by reasoning backwards from interests by EG. The role of ei-levels will be explained below. The ei-level of a skolem-constant or skolem-function is stored on its property list.

The significance of skolemization and unification is that standard rules of forwards inference can be generalized by employing unification. For instance, suppose we have inference-nodes supporting P and $(Q \rightarrow R)$, and (u_1,u_2) unifies P and Q. Then P and $(Q \rightarrow R)$ contain free variables, and so imply their own instances $u_1(P)$ and $u_2(Q \rightarrow R)$. $u_2(Q \rightarrow R) = (u_2(Q) \rightarrow u_2(R))$, and $u_1(P) = u_2(Q)$, so by *modus-ponens* we can infer $u_2(R)$. This reasoning can be shortened by rewriting *modus-ponens* as the following rule:

If (u_1,u_2) unifies P and Q, then from P and $(Q \rightarrow R)$, infer $u_2(R)$.

To illustrate, consider the following implication and non-implication:

(i) $(\exists y)(\forall x)(F\ x\ y)$, $(\exists x)(\forall y)((F\ x\ y) \rightarrow P)$ $\vDash P$

(ii) $(\forall x)(\exists y)(F\ x\ y)$, $(\forall y)(\exists x)((F\ x\ y) \rightarrow P)$ $\nvDash P$

The easiest way to see that (ii) is invalid is to note that it is equivalent to:

$(\forall x)(\exists y)(F\ x\ y)$, $((\exists y)(\forall x)(F\ x\ y) \rightarrow P)$ $\nvDash P$

but $(\forall x)(\exists y)(F\ x\ y)$ does not imply $(\exists y)(\forall x)(F\ x\ y)$.

The implication can be established easily as follows:

```
Given premises:
    (some y)(all x)(F x y)    justification = 1
    (some x)(all y)((F x y) -> P)    justification = 1
Ultimate epistemic interests:
    P    interest = 1


  # 1   (some y)(all x)(F x y)
  given
  maximal-degree-of-support: 1


  # 2   (some x)(all y)((F x y) -> P)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: P> from the inference-queue.
                                  # 1
                                  interest: P
                                  This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 3   (all x)(F x @y0)
  Inferred by support-link #1 from { 1 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.

  # 4   (F x1 @y0)
  Inferred by support-link #2 from { 3 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 5   (all y)((F @y2 y) -> P)
  Inferred by support-link #3 from { 2 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.

  # 6   ((F @y2 x3) -> P)
  Inferred by support-link #4 from { 5 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.

  # 7   P
  Inferred by support-link #5 from { 4 , 6 } by modus-ponens1
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
              ----------------------------------------
              #<Node 7> answers #<Query #1: P>
              ----------------------------------------
            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in P
  is answered affirmatively by conclusion 7
===========================================================================
ARGUMENT #1
This is a deductive argument for:
      P
 which is of ultimate interest.

 1.  (some y)(all x)(F x y)     given
 3.  (all x)(F x @y0)     EI from { 1 }
 4.  (F x1 @y0)     UI from { 3 }
 2.  (some x)(all y)((F x y) -> P)     given
 5.  (all y)((F @y2 y) -> P)     EI from { 2 }
 6.  ((F @y2 x3) -> P)     UI from { 5 }
 7.  P     modus-ponens1 from { 4 , 6 }
===========================================================================
```

However, a similar attempt to prove the non-implication fails:

```
Given premises:
     (all x)(some y)(F x y)    justification = 1
     (all y)(some x)((F x y) -> P)    justification = 1
Ultimate epistemic interests:
     P    interest = 1


  # 1   (all x)(some y)(F x y)
  given
  maximal-degree-of-support: 1


  # 2   (all y)(some x)((F x y) -> P)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: P> from the inference-queue.
                                    # 1
                                    interest: P
                                    This is of ultimate interest
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 3   (some y)(F x0 y)
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.

  # 4   (F x0 (@y1 x0))
  Inferred by support-link #2 from { 3 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 5   (some x)((F x x2) -> P)
  Inferred by support-link #3 from { 2 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.

  # 6   ((F (@y3 x2) x2) -> P)
  Inferred by support-link #4 from { 5 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.


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

The reason this fails is that the node-formula of node 4 does not unify with the antecedent of the node-formula of node 6.

Recall that substantive-reasons are instantiated by FORWARDS-MATCH-INSTANCES. In sentential reasoning, this was done by pattern-matching, but it must now be done by unification. Basically, the list of reason-premises is matched with a set of node-formulas for nodes in the list of *processed-conclusions*. This match may result in each reason-variable being matched against more than one term in the node formulas. What is required for a successful match is that all of the terms matched against a single reason-variable can be unified. Those unifications produce a list of instantiations of the node-formulas which is then passed to DRAW-CONCLUSION.

The list of instantiations is the list of substitutions produced by unification. For a monadic inference rule, this is $\langle$T$\rangle$. For a binary inference rule, this is the list of two substitutions produced by unification, which is the same thing as the unifier. For instance, if modus-ponens produces the unifier (u_1,u_2) when applied to P and $(P^* \rightarrow Q^*)$, then u_1 is the instantiation applied to P, and u_2 is that applied to $(P^* \rightarrow Q^*)$. For inferences in accordance with a ternary inference rule, the list of instantiations would be a triple, and so on. The list of instantiations is used for constructing the list of reductio-ancestors, the non-reductio-supposition, and the node-supposition for the node resulting from the forwards-inference. The actual computation of the list of reductio-ancestors, the non-reductio-supposition, and the node-supposition is done by CONCLUSION-DATA. Thus DRAW-CONCLUSION is revised as follows, where the changes are printed against a grey background:

```
DRAW-CONCLUSION P B R discharge instantiations discount (optional-arguments: binding interest)
  • 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 (CONCLUSION-DATA B P instantiations interest discharge) is nonempty, let RA be its first member(the reductio-ancestors), NRS the second member (the non-reductio-supposition), and sup the third member of Q. Let S be the sequent ⟨sup,P⟩.
  • Ifsup does not contain an explicit contradiction and not SUBSUMED S ND NRS :
    • 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 with reductio-ancestors RA and non-reductio-supposition NRS. 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, construct the set of new non-circular arguments this produces for node and its inference-descendants, 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.
```

As will be explained below, the list of reductio-ancestors and the non-reductio-supposition will become a list of pairs ⟨*node,formula*⟩ where *node* is the node recording the supposition and *formula* is the instance of the node-formula being assumed by the current inference-node. With this understanding, CONCLUSION-DATA is as follows:

CONCLUSION-DATA *basis formula instantiations interest discharge*.

- Let *RA* be the union of the results of applying each instantiation to the set of reductio-ancestors of the corresponding member of *basis* and removing any of the resulting pairs whose first members are equal to *discharge*.
- Let *NR* be the union of the results of applying each instantiation to the set of non-reductio-supposition of the corresponding member of *basis* and removing any of the resulting pairs whose first members are equal to *discharge*.
- Let *sup* be the union of the results of applying each instantiation to the set of node-suppositions of the corresponding member of *basis* and removing any of the resulting formulas that are equal to *discharge*.
- Return ⟨*RA*,*NR*,*sup*⟩.

The two main changes to DRAW-CONCLUSION are the use of CONCLUSION-DATA and the fact that SUBSUMED now takes *NRS* as an argument. The details of the latter will be discussed below. DRAW-CONCLUSION also takes *discount* as a new argument.

In addition to EI and UI, we add four forwards-logical-reasons pertaining specifically to quantifiers:

neg-UG: infer $(\exists x)\neg p$ from $\sim(\forall x)p$.

neg-EG: infer $(\forall x)\neg p$ from $\sim(\exists x)p$.

E-removal: infer $(\forall x)(p \rightarrow q)$ from $((\exists x)p \rightarrow q)$.

A-removal: infer $(\exists x)(p \rightarrow q)$ from $((\forall x)p \rightarrow q)$.

Unification in Discharging Interests

In interest-driven reasoning, we reason forwards from given premises, drawing conclusions, and backwards from interests until we find a conclusion that discharges an interest. An interest-link attaches an interest to a set of interests from which it can be derived, so when all of the link-interests of a link have been discharged, that discharges the link itself, which discharges the resultant-interest. This reasoning can also be reformulated in terms of unification. For this purpose we can skolemize interests in much the same way we did for conclusions.

Unification plays only a minor role role in backwards-reasoning. The latter begins by matching an interest-formula against the (first) conclusion of a backwards-reason. That match may result in more than one term being matched with a single reason-variable, in which case those terms must be unified with one another. The reason-variables are then instantiated by binding them to the results of the unification.

Unification plays are more substantial role when we turn to discharging interests, but it does so with a surprising twist. Consider a case in which a combination of instantiation and generalization operations should allow the discharge of an interest by a conclusion. Begin with a simple case in which the node-formula is $(F\ a)$ and the interest-formula is $(\exists y)(F\ y)$. When we strip the quantifier off the interest, we are then *looking for a binding* for the free variable. This is found by unifying $(F\ y)$ with $(F\ a)$. The important observation is that, for purposes of unification, we treat y as a variable rather than as a constant. On the other hand, if the interest is $(\forall x)(F\ x)$, and we remove the quantifier to produce $(F\ x)$, this interest should not be satisfied by the conclusion $(F\ a)$. So for purposes of unification, x should not be regarded as a variable. The interest $(F\ x)$ should only be discharged by a conclusion of the form $(F\ z)$ where z is a conclusion-variable (i.e., z is obtained by UI). What this indicates is that if interest-discharge is to be handled in terms of unification, we must "reverse-skolemize" the interests. That is, existential variables in the interest become free-variables and universal-variables become skolem-constants or terms built out of skolem-functions and free-variables. To illustrate this with a more complicated example, suppose the node-formula is $(\forall x)(\exists y)(\forall z)(F\ x\ y\ z\ b)$ and the interest-formula is $(\exists u)(\forall w)(\exists v)(F\ a\ u\ w\ v)$. Applying UI and EI to the conclusion yields $(F\ x\ (@y\ x)\ z\ b)$. Reverse-skolemizing the interest yields $(F\ a\ \hat{}@u\ \hat{}w(u)\ \hat{}@v)$. (Typographically, I will distinguish interest-variables and interest-skolem-functions from conclusion-variables and conclusion-skolem-functions by writing the former with preceding carats: $\hat{}x$ or $\hat{}@x$.) These unify to give us $(F\ a\ y\ \hat{}v\ b)$. In general, a skolemized node-formula and a reverse-skolemized interest-formula unify iff the interest can be obtained from the conclusion by a combination of universal and existential instantiation and universal and existential generalization.

To further illustrate, consider the following set of implications and non-implications:

(xi) $(\exists y)(\forall x)(F\ x\ y) \vDash (\forall x)(\exists y)(F\ x\ y)$

(xii) $(\forall x)(\exists y)(F\ x\ y) \nvDash (\exists y)(\forall x)(F\ x\ y)$

(xiii) $(\exists y)(\forall x)(F\ x\ y) \vDash (\exists y)(F\ y\ y)$

(xiv) $(\forall x)(\exists y)(F\ x\ y) \nvDash (\exists y)(F\ y\ y)$

(xv) $(\forall x)(F\ x\ x) \vDash (\forall y)(\exists x)(F\ x\ y)$

(xvi) $(\forall x)(F\ x\ x) \nvDash (\exists x)(\forall y)(F\ x\ y)$

Doing interest-discharge in terms of reverse-skolemization, these problems can be handled as follows:

```
Problem #11

Given premises:
    (some y)(all x)(F x y)    justification = 1
Ultimate epistemic interests:
    (all x)(some y)(F x y)    interest = 1


  # 1   (some y)(all x)(F x y)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (all x)(some y)(F x y)> from the inference-queue.
                                # 1
                                interest: (all x)(some y)(F x y)
                                This is of ultimate interest
                                # 2
                                interest: (some y)(F ^x0 y)
                                For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (some y)(F ^x0 y) supposing {  }> from the inference-queue.
                                # 3
                                interest: (F ^x0 ^@y1)
                                For interest 2 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 3: (F ^x0 ^@y1) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 2   (all x)(F x @y2)
  Inferred by support-link #1 from { 1 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
```

```
Retrieving #<Node 2> from the inference-queue.

  # 3   (F x3 @y2)
  Inferred by support-link #2 from { 2 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 4   (some y)(F ^x0 y)
  Inferred by support-link #3 from { 3 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 2

  # 5   (all x)(some y)(F x y)
  Inferred by support-link #4 from { 4 } by UG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 1
              ----------------------------------------
              #<Node 5> answers #<Query #1: (all x)(some y)(F x y)>
              ----------------------------------------
            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (all x)(some y)(F x y)
  is answered affirmatively by conclusion 5
=============================================================================
ARGUMENT #1
This is a deductive argument for:
      (all x)(some y)(F x y)
 which is of ultimate interest.

 1.  (some y)(all x)(F x y)     given
 2.  (all x)(F x @y2)     EI from { 1 }
 3.  (F x3 @y2)     UI from { 2 }
 4.  (some y)(F ^x0 y)     EG from { 3 }
 5.  (all x)(some y)(F x y)     UG from { 4 }
=============================================================================

*******************************************************************************************
*******************************************************************************************
Problem #12

Given premises:
     (all x)(some y)(F x y)    justification = 1
Ultimate epistemic interests:
     (some y)(all x)(F x y)    interest = 1


  # 1   (all x)(some y)(F x y)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (some y)(all x)(F x y)> from the inference-queue.
                                        # 1
                                        interest: (some y)(all x)(F x y)
                                        This is of ultimate interest
                                        # 2
                                        interest: (all x)(F x ^@y0)
                                        For interest 1 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (all x)(F x ^@y0) supposing {  }> from the inference-queue.
                                        # 3
                                        interest: (F (^x1 ^@y0) ^@y0)
                                        For interest 2 by UG
---------------------------------------------------------------------------
```

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

  # 2   (some y)(F x2 y)
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 3   (F x2 (@y3 x2))
  Inferred by support-link #2 from { 2 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Interest 3: (F (^x1 ^@y0) ^@y0) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (some y)(all x)(F x y)
  is unsatisfied.  NO ARGUMENT WAS FOUND.
==============================================================================

******************************************************************************************
******************************************************************************************
Problem #13

Given premises:
     (some y)(all x)(F x y)    justification = 1
Ultimate epistemic interests:
     (some y)(F y y)    interest = 1


  # 1   (some y)(all x)(F x y)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (some y)(F y y)> from the inference-queue.
                                        # 1
                                        interest: (some y)(F y y)
                                        This is of ultimate interest
                                        # 2
                                        interest: (F ^@y0 ^@y0)
                                        For interest 1 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (F ^@y0 ^@y0) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 2   (all x)(F x @y1)
  Inferred by support-link #1 from { 1 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 3   (F x2 @y1)
  Inferred by support-link #2 from { 2 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 4   (some y)(F y y)
  Inferred by support-link #3 from { 3 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 1
              ----------------------------------------
              #<Node 4> answers #<Query #1: (some y)(F y y)>
              ----------------------------------------
            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (some y)(F y y)
  is answered affirmatively by conclusion 4
===============================================================================
ARGUMENT #1
This is a deductive argument for:
      (some y)(F y y)
 which is of ultimate interest.

 1.  (some y)(all x)(F x y)     given
 2.  (all x)(F x @y1)     EI from { 1 }
 3.  (F x2 @y1)     UI from { 2 }
 4.  (some y)(F y y)     EG from { 3 }
===============================================================================

*******************************************************************************************
*******************************************************************************************
Problem #14

Given premises:
     (all x)(some y)(F x y)    justification = 1
Ultimate epistemic interests:
     (some y)(F y y)    interest = 1


  # 1   (all x)(some y)(F x y)
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (some y)(F y y)> from the inference-queue.
                                        # 1
                                        interest: (some y)(F y y)
                                        This is of ultimate interest
                                        # 2
                                        interest: (F ^@y0 ^@y0)
                                        For interest 1 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (F ^@y0 ^@y0) supposing {  }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 2   (some y)(F x1 y)
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 3   (F x1 (@y2 x1))
  Inferred by support-link #2 from { 2 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (some y)(F y y)
  is unsatisfied.  NO ARGUMENT WAS FOUND.
===============================================================================
******************************************************************************************
******************************************************************************************
Problem #15

Given premises:
     (all x)(F x x)    justification = 1
Ultimate epistemic interests:
     (all y)(some x)(F x y)    interest = 1


  # 1   (all x)(F x x)
  given
  maximal-degree-of-support: 1

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

  # 2   (F x0 x0)
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Query #1: (all y)(some x)(F x y)> from the inference-queue.
                                        # 1
                                        interest: (all y)(some x)(F x y)
                                        This is of ultimate interest
                                        # 2
                                        interest: (some x)(F x ^x1)
                                        For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (some x)(F x ^x1) supposing {  }> from the inference-queue.
                                        # 3
                                        interest: (F ^@y2 ^x1)
                                        For interest 2 by EG
                                        Conclusion #2 discharges interest #3

  # 3   (some x)(F x ^x1)
  Inferred by support-link #2 from { 2 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 2

  # 4   (all y)(some x)(F x y)
  Inferred by support-link #3 from { 3 } by UG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 1
               ----------------------------------------
               #<Node 4> answers #<Query #1: (all y)(some x)(F x y)>
               ----------------------------------------
             ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (all y)(some x)(F x y)
  is answered affirmatively by conclusion 4
=============================================================================
ARGUMENT #1
This is a deductive argument for:
      (all y)(some x)(F x y)
 which is of ultimate interest.

 1.  (all x)(F x x)     given
 2.  (F x0 x0)     UI from { 1 }
 3.  (some x)(F x ^x1)     EG from { 2 }
 4.  (all y)(some x)(F x y)     UG from { 3 }
============================================================================

******************************************************************************************
******************************************************************************************
Problem #16

Given premises:
     (all x)(F x x)    justification = 1
Ultimate epistemic interests:
     (some x)(all y)(F x y)    interest = 1


  # 1   (all x)(F x x)
  given
  maximal-degree-of-support: 1

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

  # 2   (F x0 x0)
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Query #1: (some x)(all y)(F x y)> from the inference-queue.
                                    # 1
                                    interest: (some x)(all y)(F x y)
                                    This is of ultimate interest
                                    # 2
                                    interest: (all y)(F ^@y1 y)
                                    For interest 1 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (all y)(F ^@y1 y) supposing {  }> from the inference-queue.
                                    # 3
                                    interest: (F ^@y1 (^x2 ^@y1))
                                    For interest 2 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 3: (F ^@y1 (^x2 ^@y1)) supposing {  }> from the inference-queue.


================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (some x)(all y)(F x y)
  is unsatisfied.  NO ARGUMENT WAS FOUND.
---------------------------------------------------

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

To implement this reasoning, we add a slot to interests for *interest-variables*, and similarly to i-lists for *i-list-variables.* The reverse skolemization of interests is accomplished by a pair of backwards-logical-reasons, EG and UG:

UG *sequent interest*

If the interest-formula *p* of *interest* is a universal generalization:

- Let e-vars be the interest-variables of interest, let *x* be the variable bound by the initial universal quantifier, and let *p** be the matrix of *p* (the formula immediately following the initial quantifier).
    - If *e-vars* is empty, let *fun* be a new variable ^@*y*. Let *term* be *fun.* If *e-vars* is nonempty, let *fun* be a new skolem-function ^@*y*, and let *term* be constructed by appending *fun* to the front of the list of *e-vars*.
    - Let *p*** be the result of substituting *term* for *x* in *p**.
    - Let *ug-condition* be the condition (applied to variables *node* and *unifier*) that

fun does not occur in the result of applying the first member of *unifier* to the node-supposition of *node* and *fun* does not occur in the second member of *unifier*.
- Where *sup* is the interest-supposition of *interest*, adopt interest in ⟨*sup*,*p***⟩ with target-interest *interest*, and let the discharge-condition of the new interest be *ug-condition*.

EG *sequent interest*
If the interest-formula *p* of *interest* is an existential generalization, let *x* be the variable bound by the initial existential quantifier and let *p** be the matrix of *p* (the formula immediately following the initial quantifier). Construct a new variable ^*x* and substitute it for *x* in *p** to produce *p***. Where *sup* is the interest-supposition of *interest*, adopt interest in ⟨*sup*,*p***⟩ with target-interest *interest*,

In sentential reasoning, interest-discharge was handled by storing the corresponding-i-list (if one existed) with each c-list and the corresponding-c-list (if one existed) with each i-list. An i-list and c-list corresponded if they had the same formula. But now that interest-discharge requires only unification rather than identity of formulas, there may be several corresponding-i-lists for each c-list (namely, all those with unifying formulas), and similarly there may be several corresponding-c-lists for each i-list. The slots in c-lists and i-lists are accordingly renamed "corresponding-i-lists" and "corresponding-c-lists". The c-list of i-list *for* a formula is now taken to be the first one whose formula whose a notational variant of the given formula. To implement this, MATCHING-C-LISTS-FOR returns the set of pairs ⟨*c-list, unifier*⟩ where *unifier* unifies the c-list-formula of *c-list* with the given formula; and similarly for MATCHING-I-LISTS-FOR. It is these lists of pairs that are stored in the slots *corresponding-c-lists* and *corresponding-i-lists*.

Interest-discharge proceeds by looking at corresponding-i-lists, and link-discharge proceeds by looking at corresponding-c-lists. Changes are printed against a grey background:

DISCHARGE-INTEREST-IN *node corresponding-i-lists old-degree new? (optional-argument: interest)*
If either *new?* is T or the maximal-degree-of-support of *node* is greater than *old-degree*:
- Let δ be the maximal-degree-of-support of *node*.
- For each pair ⟨*i-list*,*unifier*⟩ in *corresponding-i-lists*:
  - Let *sup* be the result of applying the first member of *unifier* to the node-supposition of *node*.
  - For each interest *N* in the list of i-list-interests of *i-list*, if *N* is *interest*, or (1) either *new?* is T, or δ is greater than *old-degree*, (2) the degree-of-interest of *N* is less than or equal to δ, (3) *node* satisfies the discharge-condition of *N*, and (4) APPROPRIATELY-RELATED-SUPPOSITIONS *node N unifier*, then:
  - Let *N-sup* be the result of applying the second member of *unifier* to the interest-supposition of *N*.
  - For all right-links *link* of *N*, if the degree of interest in the resultant-interest of *link* is less than or equal to δ, and either *new*? is T or the link-strength of *link* is greater than *old-degree*:
    - If the link-rule of link is "answer" (i.e., it is a link to a query), add *node* to the list of supporting-nodes of *link* and DISCHARGE-LINK *link*.
    - Otherwise, DISCHARGE-LINK *link*.
  - If the second member of *unifier* is T and *sup* is a subset of *N-sup*, then:
    - If *node* was established deductively, CANCEL-INTEREST-IN *N node*.
    - Otherwise, for any generated-suppositions *sup* of *interest* which are such that *node* is deductive in *sup*, CANCEL-NODE *sup node*.

A further complication arises from the fact that standard backwards reasoning can lead to interests that are in effect notational variants of one another, differing only by containing different free variables. If that were allowed, it would lead to a needless duplication of reasoning. To avoid this, when using CONDITION-SATISFYING-INTEREST to search for pre-existing interests, matching interests are returned along with the pattern-match that matches them against the interest-formula. The pattern-match is then stored along with the interest in the list of link-interests, and is used in computing the conclusion of a link-discharge.

When an interest-link is constructed in response to adopting an interest, the interest-formula is matched to the reason-conclusion to produce a binding for the reason-variables. The link-interest is produced by applying the binding to the first backwards-premise. When a node is found that unifies with that interest, if there are further backwards-premises, then a new link is constructed. In computing the link-interest for the new link, we must apply the unifier to the earlier binding. In this way the binding is systematically modified as the link-interests are discharged. When there are no remaining-premises, the conclusion to be drawn is computed by applyng the binding to the reason-conclusion. To accommodate all of this, a new slot is added to interest-links to hold the binding. We also add a slot to hold the list of instantiations for the conclusion-variables in the supporting-nodes of the link. APPROPRIATELY-RELATED-SUPPOSITIONS (discussed below) needs to refer to the interest-supposition that is produced from the supposition of the resultant-interest by applying the sequentially produced unifiers, so a slot is added to interest-links to hold the link-supposition as well. MERGE-UNIFIERS* merges two unifiers to produce a single unifier which is such that applying it to a formula is equivalent to applying the first-unifier first and then the second unifier. Then DISCHARGE-LINK is modified as follows:

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

- For each pair (*c-list unifier*) in a corresponding-c-list of the interest-i-list of the resultant-interest of *link*, for each inference-node *N* in the c-list-nodes of *c-list*, 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, let *unifiers* be the result of applying APPROPRIATELY-RELATED-SUPPOSITIONS to *N*, the link-interest of *link*, the link-supposition of *link*, the interest-supposition-variables of the link-interest of *link*, and *unifier*. If *unifiers* is nonempty then for each *u* in *unifiers*:
  - Let *u** be the result of applying MERGE-UNIFIERS* to *unifier* and *u*.
  - Let *binding* be the result of applying the second member of *u** to the link-binding of *link*.
  - Let *instantiations* be the result of adding the first member of *u** to the front of the list produced by applying the second member of *u** to each member of the link-instantiations of *link*.
  - Let *supposition* be the result of applying the second member of *u** to the link-supposition of *link*.
  - 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 *formulas* be the result of applying the link-binding to the conclusions of the link-reason, and for each *formula* in *formulas*, apply DRAW-CONCLUSION to*formula*, the result of adding *N* to the list of supporting-nodes of *link, R*, *instantiations*, the discount-factor of *R*, *R discharge*, *bninding*, and the resultant-interest of *link*.
  - If instead *link* has remaining-premises, construct a new interest-link by inserting *N* 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, setting the link-instantiations to *instantiations*, the link-binding to *binding*, and the link-supposition to *supposition.* Apply DISCHARGE-LINK to the new-link and δ.
  - If *N* is a deductive-node, the second member of *unifier* is T, and result of applying the first member of *unifier* to the node-supposition of *N* is a subset of the result of applying the second member of *unifier* to the interest-

supposition of *interest*, CANCEL-INTEREST-IN *interest*. Otherwise, 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.

The cancellation of interests discharged by deductive-nodes is now more complex than it was when formulas did not contain free variables. There is no guarantee that such a deductive-node will enter into drawing a conclusion, because if the reason involved has multiple premises then inference-nodes could unify with the individual premises without unifying collectively with the set of premises. We are only guaranteed to be able to use the deductive node in drawing whatever conclusion may eventually be drawn when either the interest-sequent contains no free-variables (so only trivial unification is involved in the discharge) or the reason-length of the link-rule is 1 so that the unifier need not be consistent with other unifiers for other members of the link-basis. Accordingly, this is the only case in which the link-interest can be cancelled.

Suppositional reasoning

Suppositional reasoning introduces new complexities. There are two kinds of suppositional reasoning that occur in OSCAR — conditionalization, and reductio. Let us begin with conditionalization. Given a reverse-skolemized interest $(P \rightarrow Q)$, we want to find a conclusion $(P^* \rightarrow Q^*)$ that unifies with it. Such a conclusion can be obtained by conditionalization from a conclusion $Q^*/\{P^*\}$. But notice that the latter conclusion unifies with $Q/\{P\}$ iff $(P^* \rightarrow Q^*)$ unifies with $(P \rightarrow Q)$. Thus we can direct the search for $(P^* \rightarrow Q^*)$ by looking for something that unifies with $Q/\{P\}$, and then conditionalizing. This is done by adopting interest in $Q/\{P\}$ and reasoning backwards. To get something that unifies with $Q/\{P\}$, we must make a supposition $P^*/\{P^*\}$ that unifies with $P/\{P\}$, and then draw the corresponding conclusion Q^* with respect to $\{P^*\}$. Thus we must look for an appropriate supposition $P^*/\{P^*\}$ to make. This in turn can be done by making the supposition $P/\{P\}$, and treating it as a schematic supposition. That is, we treat the interest-variables in P as schematic variables, and instantiate them in any way that might lead us to a conclusion of the form $Q^*/\{P^*\}$. This is analogous to the way we treat conclusion-variables in forwards reasoning. That is, we instantiate them in any way that is potentially useful, and our test for that is that by instantiating them in that way we can make a forwards inference. So we adopt the supposition $P/\{P\}$, and reason forwards from it, treating the interest-variables just like conclusion-variables for purposes of forwards reasoning. This will be done by including them among the node-variables for the node encoding the supposition.

In sentential reasoning, before making a new supposition we check that the same supposition has not already been made. In first-order reasoning, we must check more generally that no supposition has been made whose schematic variables can be instantiated in such a way as to make the new supposition an instance of it.

Given an interest in $(P \rightarrow Q)/X$, OSCAR will adopt interest in $Q/X\cup\{P\}$. Then a conclusion Q^*/Y^* should discharge this interest iff there is a subset (appropriately ordered) X_0 of $X\cup\{P\}$ such that Q^*/Y^* unifies with Q/X_0. Recall that a unifier *u unifies* a list of expressions X *into* a list of expressions Y relative to the variable-lists *var1* and *var2* iff there is an appropriately-ordered sublist Y_0 of Y such that u unifies X with Y relative *var1* and *var2*. The function SET-UNIFIER, defined above, finds all such unifiers. Let (MERGE-MATCHES* *m1 m2*) be a match equivalent to applying *m1* first and then *m2*. Then interest-discharge can proceed as follows. First determine whether Q^* unifies with Q via some unifier $(u_1\ u_2)$. If so, apply the unifiers u_1 and u_2 to the suppositions Y^* and $X\cup\{P\}$, producing $u_1(Y^*)$ and $u_2(X\cup\{P\})$. If $u_1(Y^*) \subseteq u_2(X\cup\{P\})$, that is sufficient for the conclusion to discharge the interest, but this is not a necessary condition for interest-discharge. The difficulty is that Y^* may contain free variables not occurring in Q^* and $X\cup\{P\}$ may contain free variables not occurring in X. In that case, interest-discharge can still occur if there is a unifier $(u_1^*\ u_2^*)$ that unifies $u_1(Y^*)$ into $u_2(X\cup\{P\})$ relative to those remaining variables. If there is such a unifier, then if u_1^{**} = (MERGE-MATCHES* $u_1\ u_1^*$) and u_2^{**} = (MERGE-MATCHES* $u_2\ u_2^*$), the conclusion Q^*/Y^* should discharge the interest $Q/X\cup\{P\}$ relative to the unifer

(u_1** u_2**). To illustrate, consider the following argument for $(\exists x)(\exists y)((F\ x\ y) \rightarrow (F\ x\ x))$:

```
Given premises:
Ultimate epistemic interests:
    (some x)(some y)((F x y) -> (F x x))    interest = 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (some x)(some y)((F x y) -> (F x x))> from the inference-queue.
                                  # 1
                                  interest: (some x)(some y)((F x y) -> (F x x))
                                  This is of ultimate interest
                                  # 2
                                  interest: (some y)((F ^@y0 y) -> (F ^@y0 ^@y0))
                                  For interest 1 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 2: (some y)((F ^@y0 y) -> (F ^@y0 ^@y0)) supposing {  }> from the inference-queue.
                                  # 3
                                  interest: ((F ^@y0 ^@y1) -> (F ^@y0 ^@y0))
                                  For interest 2 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 3: ((F ^@y0 ^@y1) -> (F ^@y0 ^@y0)) supposing {  }> from the inference-queue.

  # 1   (F ^@y0 ^@y1)    supposition: { (F ^@y0 ^@y1) }
  supposition
  maximal-degree-of-support: 1

                                  # 4
                                  interest: (F ^@y0 ^@y0)    supposition: { (F ^@y0 ^@y1) }
                                  For interest 3 by conditionalization
                                  Conclusion #1 discharges interest #4

  # 2   ((F ^@y0 ^@y0) -> (F ^@y0 ^@y0))
  Inferred by support-link #1 from { 1 } by conditionalization
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 3

  # 3   (some y)((F ^@y1 y) -> (F ^@y1 ^@y1))
  Inferred by support-link #2 from { 2 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 2

  # 4   (some x)(some y)((F x y) -> (F x x))
  Inferred by support-link #3 from { 3 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 1
            ----------------------------------------
            #<Node 4> answers #<Query #1: (some x)(some y)((F x y) -> (F x x))>
            ----------------------------------------
           ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
  Interest in (some x)(some y)((F x y) -> (F x x))
  is answered affirmatively by conclusion 4
===========================================================================
ARGUMENT #1
This is a deductive argument for:
     (some x)(some y)((F x y) -> (F x x))
 which is of ultimate interest.

        |----------------------------------------------------------
        | Suppose:  { (F ^@y0 ^@y1) }
        |----------------------------------------------------------
        | 1.  (F ^@y0 ^@y1)     supposition
```

```
2.  ((F ^@y0 ^@y0) -> (F ^@y0 ^@y0))     conditionalization from { 1 }
3.  (some y)((F ^@y1 y) -> (F ^@y1 ^@y1))     EG from { 2 }
4.  (some x)(some y)((F x y) -> (F x x))     EG from { 3 }
=========================================================================
```

In this example, conclusion 1, (F ^@y0 ^@y1) / { (F ^@y0 ^@y1) }, discharges interest in the sequent (F ^@y0 ^@y0)/{ (F ^@y0 ^@y1) }. Unifying (F ^@y0 ^@y1) with (F ^@y0 ^@y0) produces the unifier (((^@y1 . ^@y0)) t), and applying that to the suppositions yields { (F ^@y0 ^@y0) } and { (F ^@y0 ^@y1) }. The former is not a subset of the latter, but it does unify into it via the unifier (t ((^@y1 . ^@y0))). Accordingly, (F ^@y0 ^@y1) / { (F ^@y0 ^@y1) }, discharges interest in the sequent (F ^@y0 ^@y0)/{ (F ^@y0 ^@y1) via the unifier (((^@y1 . ^@y0)) ((^@y1 . ^@y0))).

To implement conditionalization in this form, we will keep track of the interest-variables occuring in the node-supposition and interest-supposition in the slots *node-supposition-variables* and *interest-supposition-variables*, and modify the definition of APPROPRIATELY-RELATED-SUPPOSITIONS so that it unifies the node-supposition into the interest-supposition and returns the list of unifiers. Those unifiers are then used in computing the conclusion to be drawn.

APPROPRIATELY-RELATED-SUPPOSITIONS *node interest supposition unifier*

- Let *i-sup* be the result of applying the second member of *unifier* to the interest-supposition of *interest*, *n-vars* the result of applying the first member of *unifier* to the list of free-variables occurring in the node-supposition of *node* but not the node-formula of *node*, and *i-vars* the result of applying the second member of *unifier* to the list of free-variables occurring in the interest-supposition of *interest* but not the interest-formula of *interest*.
- If *interest* is a reductio-interest, let *r-sup* be the result of applying the first member of *unifier* to the list of node-formulas of those non-reductio-suppositions of *node* not contained in *inherited-non-reductio-suppositions*, and return SET-UNIFIER *r-sup i-sup n-vars i-vars*.
- Otherwise, let *n-sup* be the result of applying the first member of *unifier* to the node-supposition of *node*, and return SET-UNIFIER *n-sup i-sup n-vars i-vars*.

The presence of free variables in suppositions makes it more complicated to keep track of the non-reductio-supposition (and also the reductio-ancestors) of a node. We begin by making a supposition, and recording it in an inference-node. Reasoning forwards from that supposition can have the effect of instantiating some of the free variables in it, so sequents derived from a supposition node may contain formulas in their sequent suppositions that are instances of the original supposition formula rather being identical with the original supposition formula. For some purposes, we will subsequently want to know what inference-nodes record suppositions upon which a later inference-node depends, and for other purposes we will want to know what the supposition-formulas of an inference-node are. To record all of this information, we will take the non-reductio-supposition for a node to be a list of pairs ⟨*node,formula*⟩ where *node* is the node recording the supposition and *formula* is the instantiation of the original supposition formula that is being supposed by the current node. The list of reductio-ancestors of a node will be treated similarly.

Reductio-ad-absurdum

When we turn to reductio reasoning, the presence of free variables introduces three complications. First, reductio-interests, like other interests, will be discharged by unification and by unifying the node-supposition into the interest-supposition. In sentential reasoning, a node discharges a reductio-interest if the node-formula and the interest-formula are the same and the the non-inherited part of the non-reductio-supposition of the node is contained in the interest-supposition. With free variables, the requirement is instead that the node-formula unifies with the interest-formula via some unifier *unifier*, and then the result of applying the first member of *unifier* to the non-inherited part of the non-reductio-supposition of the node unifies into the result of applying the second member of *unifier* to the interest-supposition. This is the way the definition of APPROPRIATELY-RELATED-SUPPOSITIONS was revised above.

DISCHARGE-REDUCTIOS *node i-lists*

- DISCHARGE-FORTUITOUS-REDUCTIOS *node*
- If *node* is has reductio-ancestors, then for every pair ⟨*i-list*,*unifier*⟩ in *i-lists*, let *Y* be the result of applying the first member of *unifier* to the node-supposition of *node*. For each direct-reductio-interest *interest* in the list of i-list-interests of *i-list*, let *unifiers* be the list of unifiers produced by APPROPRIATELY-RELATED-SUPPOSITIONS *node interest unifier*:
    - 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 *node**) and for each *unifier** produced by first applying *unifier* and then a member of *unifiers*:
        - Let *Y** be the result of applying the second member of *unifier** to the node-supposition of *node**, *RA* the union of the result of applying the first member of *unifier** to the reductio-ancestors of *node* and the result of applying the second member of *unifier** to the reductio-ancestors of *node**, and *NR* the union of the result of applying the first member of *unifier** to the non-reductio-supposition of *node* and the result of applying the second member of *unifier** to the non-reductio-supposition of *node**.
        - For each *R* in *RA*, if either *R* is not a base-reductio-supposition or it is the only member of *RA*, and the node-formula of *R* is *P*, then DRAW-REDUCTIO-CONCLUSION *P node node* R Y Y* RA NR interest*.

DISCHARGE-RETROSPECTIVE-REDUCTIOS *node. interest*

For every pair ⟨*c-list*,*unifier*⟩ in the list of corresponding-c-lists of the interest-i-list of *interest*, let *Y** be the result of applying the second member of *unifier* to the node-supposition of *node*. For each node *N* in the list of c-list-nodes of *c-list*, if the maximal-degree-of-support of *N* is greater than or equal to the degree-of-interest of *interest*, let *unifiers* be the list of unifiers produced by APPROPRIATELY-RELATED-SUPPOSITIONS *node interest unifier*. For each *unifier** produced by first applying *unifier* and then a member of *unifiers*:

- Let *Y* be the result of applying the first member of *unifier** to the node-supposition of *N*, *RA* the union of the result of applying the first member of *unifier** to the reductio-ancestors of *N* and the result of applying the second member of *unifier** to the reductio-ancestors of *node*, and *NR* the union of the result of applying the first member of *unifier** to the non-reductio-supposition of *N* and the result of applying the second member of *unifier** to the non-reductio-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* RA NR interest*.

The c-list-contradictors of a c-list must now be those whose formulas unify with the formula of the given c-list, so we will store pairs ⟨*contradicting-c-list*,*unifier*⟩ in the c-list-contradictors slot for a c-list. With this in mind:

DISCHARGE-FORTUITOUS-REDUCTIOS *node*

For each pair ⟨*c-list*,*unifier*⟩ in the list of c-list-contradictors of the c-list of *node*, and for each *node** in the list of c-list-nodes of *c-list*,

- If *node* and *node** both have empty node-suppositions, then for each ultimate-epistemic-interest, let *P* be the result of applying the first member of *unifier* to the query-formula, and infer *P* 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 relative to the trivial unifier '(T T), then where *P* is the result of applying the first member of *unifier* to the interest-formula of *in*, DRAW-CONCLUSION *P* {*node*, *node**} "fortuitous-reductio" *unifier* 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 relative to the trivial unifier '(T T), then where *P* is the result of applying the second member of *unifier* to the interest-formula of *in* and *unifier** is the result of reversing the order of the members of *unifier*, DRAW-CONCLUSION *P* {*node*, *node*} "fortuitous-reductio"*unifier** NIL.

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

If neither *node* nor *node** is a cancelled-node, ¬*P* is not in *Y* or *Y**, and neither *RA* nor *NR* contains multiple instantiations of the same supposition-node unless that supposition was made more than once, then:

- Let *sup* be *Y*∪*Y**-{*P*}, *S* the sequent ⟨*sup*,¬*P* ⟩, *reductio-ancestors* the result of removing *R* from *RA*, *non-reductio-supposition* the subset of *NR* consisting of pairs whose first members are not equal to *P*, and *NDA* the union of the crossproduct of the nearest-defeasible-ancestors of *node* and the nearest-defeasible-ancestors of *node**. If not SUBSUMED *S NDA non-reductio-supposition*:
  - If there is an inference-node of kind "inference" supporting *S* and having *non-reductio-supposition* as its list of non-reductio-suppositions and *reductio-ancestors* as its list of reductio-ancestors, 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*. Add the members of *NDA* to the list of nearest-defeasible-ancestors for *conclusion*, and set the list of non-reductio-suppositions equal to *non-reductio-supposition*, and the list of reductio-ancestors equal to *reductio-ancestors*.
    - 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

conclusion 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*.

The second complication that pertains to making reductio-suppositions is the same as that required for non-reductio-suppositions. A reductio-supposition supposes the negation of a reductio-interest. As before, the conclusion-variables for the supposition will be the interest-variables for the interest. This is justified by noting that the reductio-reasoning generated by an interest in P is analogous to adopting interest in the conditionals $(\sim P \rightarrow (Q \,\&\, \sim Q))$ for different choices of Q, and then trying to establish these conditionals by conditionalization. We do the latter by supposing $\sim P$, and taking the conclusion-variables of the supposition to be the interest-variables of the interest. And, as observed above, in sentential reasoning, before making a new supposition we check that the same supposition has not already been made. In first-order reasoning, we must check more generally that no supposition has been made whose schematic variables can be instantiated in such a way as to make the new supposition an instance of it.

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 reductio-
  supposition supposing q,, let that be sup. If not, but there is an existing reductio-
  supposition whose node-formula matches q relative to its node-variables, let that
  be sup. If there is no such reductio-supposition, but there is an existing non-
  reductio-supposition whose node-formula matches q relative to its node-
  variables, let that be sup, and 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 $\neg q/\{\neg q\}$, whose
      node-variables are the interest-variables of interest, 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.
```

The third complication concerns the generation of reductio-interests from conclusions containing free variables. To take a simple example, suppose we have a conclusion $(\forall x)(F\ x)$, and by UI we infer $(F\ x)$. To use this conclusion in reductio-reasoning, we will adopt interest in its negation $\sim(F\ x)$. In the resulting interest, how is the free variable x to be treated? In the conclusion $(F\ x)$, x is a conclusion variable, unifying with any term. What we want for reductio-reasoning is to find *some* instance of $\sim(F\ x)$. What this means is that in the interest in $\sim(F\ x)$, x is to be treated as an interest-variable.

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 supposi-
  tion 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. Let the interest-
  variables of the new interest be the node-variables 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*.

To illustrate all of this, consider a simple example of reductio-reasoning using free variables:

```
Given premises:
    (all x)((P x) <-> ((H x) & ~(P x)))    justification = 1
Ultimate epistemic interests:
    (all x)~(H x)    interest = 1


  # 1   (all x)((P x) <-> ((H x) & ~(P x)))
  given
  maximal-degree-of-support: 1

---------------------------------------------------------------------------
Retrieving #<Query #1: (all x)~(H x)> from the inference-queue.
                                  # 1
                                  interest: (all x)~(H x)
                                  This is of ultimate interest
                                  # 2
                                  interest: ~(H ^x0)
                                  For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 2: ~(H ^x0) supposing {  }> from the inference-queue.

  # 2   (H ^x0)    supposition: { (H ^x0) }
  reductio-supposition
  maximal-degree-of-support: 1

                              Readopting interest in:
                                  # 2
                                  reductio interest: ~(H ^x0)
                                  For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Node 1> from the inference-queue.

  # 3   ((P x1) <-> ((H x1) & ~(P x1)))
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.

  # 4   ((P x1) -> ((H x1) & ~(P x1)))
  Inferred by support-link #2 from { 3 } by bicondit-simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 5   (((H x1) & ~(P x1)) -> (P x1))
  Inferred by support-link #3 from { 3 } by bicondit-simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.

  # 6   ((H x1) -> (~(P x1) -> (P x1)))
  Inferred by support-link #4 from { 5 } by exportation
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
                                  # 3
                                  reductio interest: ~(((H x1) & ~(P x1)) -> (P x1))
                                  using node 5
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.
```

```
                                        # 4
                                        reductio interest: ~((H x1) -> (~(P x1) -> (P x1)))
                                        using node 6
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
                                        # 5
                                        reductio interest: ~((P x1) -> ((H x1) & ~(P x1)))
                                        using node 4
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.

  # 7   (~(P ^x0) -> (P ^x0))    supposition: { (H ^x0) }
  Inferred by support-link #5 from { 6 , 2 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
                                        # 6
                                        reductio interest: ~(~(P ^x0) -> (P ^x0))
                                        using node 7
---------------------------------------------------------------------------
Retrieving #<Interest 6: ~(~(P ^x0) -> (P ^x0)) supposing {  }> from the inference-queue.
                                        # 7
                                        reductio interest: ~(P ^x0)
                                        For reductio interest 6 by i-neg-condit
---------------------------------------------------------------------------
Retrieving #<Interest 7: ~(P ^x0) supposing {  }> from the inference-queue.

  # 8   (P ^x0)    supposition: { (P ^x0) }
  reductio-supposition
  maximal-degree-of-support: 1

                                    Readopting interest in:
                                        # 7
                                        reductio interest: ~(P ^x0)
                                        For reductio interest 6 by i-neg-condit
---------------------------------------------------------------------------
Retrieving #<Interest 5: ~((P x1) -> ((H x1) & ~(P x1))) supposing {  }> from the inference-queue.
                                        # 8
                                        reductio interest: ~((H x1) & ~(P x1))
                                        For reductio interest 5 by i-neg-condit
---------------------------------------------------------------------------
Retrieving #<Interest 8: ~((H x1) & ~(P x1)) supposing {  }> from the inference-queue.
                                        # 9
                                        reductio interest: (~(H x1) v (P x1))
                                        For reductio interest 8 by i-DM
---------------------------------------------------------------------------
Retrieving #<Interest 9: (~(H x1) v (P x1)) supposing {  }> from the inference-queue.
                                        # 10
                                        reductio interest: ((H x1) -> (P x1))
                                        For reductio interest 9 by disj-cond
                                        # 11
                                        reductio interest: (~(P x1) -> ~(H x1))
                                        For reductio interest 9 by disj-cond-2
---------------------------------------------------------------------------
Retrieving #<Interest 10: ((H x1) -> (P x1)) supposing {  }> from the inference-queue.

  # 9   (H x1)    supposition: { (H x1) }
  supposition
  maximal-degree-of-support: 1

                                        # 12
                                        reductio interest: (P x1)    supposition: { (H x1) }
                                        For reductio interest 10 by conditionalization
                                        Conclusion #8 discharges interest #12

  # 10   ((H ^x0) -> (P ^x0))    supposition: { (P ^x0) }
```

```
 Inferred by support-link #6 from { 8 } by conditionalization
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 10

 # 11   (~(H ^x0) v (P ^x0))    supposition: { (P ^x0) }
 Inferred by support-link #7 from { 10 } by disj-cond
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 9

 # 12   ~((H ^x0) & ~(P ^x0))    supposition: { (P ^x0) }
 Inferred by support-link #8 from { 11 } by i-DM
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 8
                              # 13
                              reductio interest: (P x1)
                              For reductio interest 5 by i-neg-condit
                              Conclusion #8 discharges interest #13

 # 13   ~((P ^x0) -> ((H ^x0) & ~(P ^x0)))    supposition: { (P ^x0) }
 Inferred by support-link #9 from { 8 , 12 } by i-neg-condit
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 5

Node #13 discharges reductio-interest #5, generated by node #4

 # 14   ~(P ^x0)
 Inferred by support-link #10 from { 13 , 4 } by reductio
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
                          Readopting interest in:
                              # 7
                              reductio interest: ~(P ^x0)
                              For reductio interest 6 by i-neg-condit using conclusion 14
                              Conclusion #14 discharges interest #7

 # 15   ~(~(P ^x0) -> (P ^x0))
 Inferred by support-link #11 from { 14 , 14 } by i-neg-condit
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 6
..........................................................................................................................
. . . . . . . . . . . . . . . . . . Cancelling  #<Interest 6: ~(~(P ^x0) -> (P ^x0)) supposing {  }>
..........................................................................................................................
..........................................................................................................................
. . . . . . . . . . . . . . . . Cancelling  #<Interest 7: ~(P ^x0) supposing {  }>
..........................................................................................................................
---------------------------------------------------------------------------
Retrieving #<Node 14> from the inference-queue.

 # 16   (P ^x0)    supposition: { (H ^x0) }
 Inferred by support-link #12 from { 7 , 14 } by modus-ponens2
 maximal-degree-of-support: 1
 This node encodes a deductive argument.

 # 17   ((H ^x0) -> (P ^x0))
 Inferred by support-link #13 from { 16 } by conditionalization
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 10

 # 18   (~(H ^x0) v (P ^x0))
 Inferred by support-link #14 from { 17 } by disj-cond
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 9

 # 19   ~((H ^x0) & ~(P ^x0))
 Inferred by support-link #15 from { 18 } by i-DM
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 8

 # 20   ~((P ^x0) -> ((H ^x0) & ~(P ^x0)))    supposition: { (H ^x0) }
 Inferred by support-link #16 from { 19 , 16 } by i-neg-condit
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 5
```

Node #20 discharges reductio-interest #5, generated by node #4

# 21   ~(H ^x0)
Inferred by support-link #17 from { 20 , 4 } by reductio

```
 maximal-degree-of-support: 1
 This node encodes a deductive argument.

 # 22   (all x)~(H x)
 Inferred by support-link #18 from { 21 } by UG
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
 This node discharges interest 1
             ----------------------------------------
             #<Node 22> answers #<Query #1: (all x)~(H x)>
             ----------------------------------------
            ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================== ULTIMATE EPISTEMIC INTERESTS ===================
 Interest in (all x)~(H x)
 is answered affirmatively by conclusion 22
==========================================================================
ARGUMENT #1
This is a deductive argument for:
     (all x)~(H x)
 which is of ultimate interest.

 1.  (all x)((P x) <-> ((H x) & ~(P x)))     given
 3.  ((P x1) <-> ((H x1) & ~(P x1)))     UI from { 1 }
 4.  ((P x1) -> ((H x1) & ~(P x1)))     bicondit-simp from { 3 }
 5.  (((H x1) & ~(P x1)) -> (P x1))     bicondit-simp from { 3 }
 6.  ((H x1) -> (~(P x1) -> (P x1)))     exportation from { 5 }
       |--------------------------------------------------------
       | Suppose:  { (P ^x0) }
       |--------------------------------------------------------
       | 8.  (P ^x0)     reductio-supposition
       | 10.  ((H ^x0) -> (P ^x0))     conditionalization from { 8 }
       | 11.  (~(H ^x0) v (P ^x0))     disj-cond from { 10 }
       | 12.  ~((H ^x0) & ~(P ^x0))     i-DM from { 11 }
       | 13.  ~((P ^x0) -> ((H ^x0) & ~(P ^x0)))     i-neg-condit from { 8 , 12 }
 14.  ~(P ^x0)     reductio from { 13 , 4 }
       |--------------------------------------------------------
       | Suppose:  { (H ^x0) }
       |--------------------------------------------------------
       | 2.  (H ^x0)     reductio-supposition
       | 7.  (~(P ^x0) -> (P ^x0))     modus-ponens2 from { 6 , 2 }
       | 16.  (P ^x0)     modus-ponens2 from { 7 , 14 }
 17.  ((H ^x0) -> (P ^x0))     conditionalization from { 16 }
 18.  (~(H ^x0) v (P ^x0))     disj-cond from { 17 }
 19.  ~((H ^x0) & ~(P ^x0))     i-DM from { 18 }
       |--------------------------------------------------------
       | Suppose:  { (H ^x0) }
       |--------------------------------------------------------
```

```
      | 20.  ~((P ^x0) -> ((H ^x0) & ~(P ^x0)))     i-neg-condit from { 19 , 16 }
  21.  ~(H ^x0)     reductio from { 20 , 4 }
  22.  (all x)~(H x)     UG from { 21 }
============================================================================
```

```
Cumulative size of arguments = 20
Size of inference-graph = 22 of which 1 were unused suppositions.
90% of the inference-graph was used in the argument.
```

In this example, node 2 is a reductio-supposition generated by interest 2, and accordingly the node-variables of node 2 are the interest-variables of interest 2, viz., ^x0. Similarly, interest 5 is a reductio-interest generated by node 4, and hence the interest-variables of interest 5 are the node-variables of node 4, viz., x1. The interest-formula of interest 5 is ~((P x1) -> ((H x1) & ~(P x1))), and that unifies with the node-formula of node 13, which is ~((P ^x0) -> ((H ^x0) & ~(P ^x0))). Thus we get a reductio-inference at node 14. Exactly the same reasoning generates a second reductio-inference at node 14.

Miscellany

There remain several details to be explained. First, the cancellation of subsumed-nodes must now proceed by matching new conclusions with previous more general conclusions. To determine whether an existing node *node* should prevent the construction of a new node supporting a sequent *S* with nearest-defeasible-ancestors *NDA* and non-reductio-supposition *non-reductio-supposition*, we must first pattern-match the sequent-formula of *S* with the node-formula of *node*, using the node-variables of the latter as the variables for pattern matching. If there is such a match *m*, let *sup* be the result of applying *m* to the node-supposition of *node*. The node-supposition may contain variables not occurring free in the node-formula. If so, find all ways of pattern-matching the sequent-supposition of *S* with subsets of *sup*. If there are no such variables, this returns the identity-match if the sequent-supposition is a subset of *sup*, and it returns no matches otherwise. Subsumption requires that there be some such match. However, that is not sufficient for subsumption. The difficulty is that when reasoning with free variables, the same supposition-formula could be produced by instantiating the node-formulas of different supposition-nodes, and one of those supposition-nodes might be a reductio-supposition while the other is not. If *S* is identical with the node-sequent of *node*, but one of its supposition-formulas is produced by a reductio-supposition-node where that same supposition-formula is produced in *node* by a non-reductio-supposition-node, then reductio-reasoning would be possible from *S* that is not possible from *node*. If, in this case, we allow the construction of a new node to be blocked by subsumption from *node*, this will block the reductio-reasoning, so this should not be a case of subsumption. Accordingly, there must be a further requirement for subsumption that for one of the matches *m** matching the sequent-supposition into *sup*, the result of applying first *m* and then *m** to the non-reductio-supposition of *node* should produce a subset of *non-reductio-supposition*. Putting this all together, we obtain the following rules for subsumption:

(a) If L_1 and L_2 are non-defeasible support-links for the same node, then L_1 subsumes L_2 if every set *X* in the list of nearest-defeasible-link-ancestors for L_2 contains as a subset some set *Y* in the list of nearest-defeasible-link-ancestors for L_1.

(b) Let N_1 be the support-link-target of L_1 and N_2 the support-link-target for L_2. If N_1 and N_2 have the same node-formula, or if their node-formulas can be matched by rewriting free-variables (by some pattern-match *m*), let *sup* be the result of applying *m* to the node-supposition of N_1. Let *SM* be the set of ways of matching *sup* with a subset of the node-supposition of N_2. If some member of *SM* is such that applying it to the non-reductio-supposition of N_1 produces a subset of the non-reductio-supposition of N_2, and either

(i) L_2 records a defeasible inference, L_1 records an infrence in accordance with the same inference-rule, the union of the nearest-defeasible-ancestors of the basis of L_1 is the same as the union of the nearest-defeasible-ancestors of the basis of L_2 (so that they will be defeated by the same defeaters), and the result of applying *m* to the binding employed in inferring L_2 equals the binding employed in L_1 (so the defeaters are the same except possibly for L_2 having defeaters containing additional formulas in their suppositions); or

(ii) L_2 records a non-defeasible inference and every set *X* in the list of nearest-defeasible-link-ancestors for L_2 contains as a subset some set *Y* in the list of nearest-defeasible-link-ancestors for L_1,

then L_1 subsumes L_2.

The *node-discount-factor* has been added as a new slot in inference-nodes, and this is used in prioritizing reasoning. COMPUTE-PRIORITIES is modified to compute the discounted-node-strength of a node as the product of its maximal-degree-of-support and its node-discount-factor. The only inference-rule in connection with which the node-discount-factor is currently used is EI. This answers to a difficulty that makes the use of EI potentially explosive. EI was formulated as follows:

```
EI node
If node is of type "inference", and its node-formula p is an existential generalization:
•  Let x be the variable bound by the initial existential quantifier, let p* be the
   matrix of p (the formula immediately following the initial quantifier), let u-vars
   be the list of node-variables of node, and let s-funs be the list of skolem-functions
   occurring in p*.
•  If s-funs is empty, let level be 1.  Otherwise, let level be one plus the maximum of
   the ei-levels of the members of s-funs.  Let discount be $.5^{(level - 1)}$.
   •  If u-vars is empty, let fun be a new variable @y.  Let term be fun  If u-vars is
      nonempty, let fun be a new skolem-function @y, and let term be be construct-
      ed by appending @y to the front of the list of u-vars.
   •  Let p** be the result of substituting term for x in p*.
   •  If there is no node node* in processed-conclusions whose node-formula differs
      from p** only by having some other singular term in place of term:
      •  set the ei-level of fun to be level
      •  return the list consisting of the single list ⟨p**, {node}, EI, ⟨T⟩, discount⟩
```

This has the effect of systematically discounting nodes generated by repetitive uses of EI. Consider its use in the following example:

```
Given premises:
    (all x)((F x) -> (some y)((F y) & (G x y)))    justification = 1
Ultimate epistemic interests:
    (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))    interest = 1


 # 1   (all x)((F x) -> (some y)((F y) & (G x y)))
 given
 maximal-degree-of-support: 1

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

 # 2   ((F x0) -> (some y)((F y) & (G x0 y)))
 Inferred by support-link #1 from { 1 } by UI
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Query #1: (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))> from the inference-queue.
                        # 1
```

```
                                interest: (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))
                                This is of ultimate interest
                                # 2
                                interest: ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z)))
                                For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 2: ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z))) supposing {  }> from the
inference-queue.

  # 3   (F ^x5)    supposition: { (F ^x5) }
  supposition
  maximal-degree-of-support: 1

                                # 3
                                interest: (some y)(some z)((G ^x5 y) & (G y z))    supposition: { (F ^x5) }
                                For interest 2 by conditionalization
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.

  # 4   (some y)((F y) & (G ^x5 y))    supposition: { (F ^x5) }
  Inferred by support-link #2 from { 2 , 3 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.

  # 5   ((F @y6) & (G ^x5 @y6))    supposition: { (F ^x5) }
  Inferred by support-link #3 from { 4 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.

  # 6   (F @y6)    supposition: { (F ^x5) }
  Inferred by support-link #4 from { 5 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 7   (G ^x5 @y6)    supposition: { (F ^x5) }
  Inferred by support-link #5 from { 5 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.

  # 8   (some y)((F y) & (G @y6 y))    supposition: { (F ^x5) }
  Inferred by support-link #6 from { 2 , 6 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 8> from the inference-queue.

  # 9   ((F @y7) & (G @y6 @y7))    supposition: { (F ^x5) }
  Inferred by support-link #7 from { 8 } by EI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Interest 3: (some y)(some z)((G ^x5 y) & (G y z)) supposing { (F ^x5) }> from the inference-
queue.
                                # 4
                                interest: (some z)((G ^x5 ^@y8) & (G ^@y8 z))    supposition: { (F ^x5) }
                                For interest 3 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 4: (some z)((G ^x5 ^@y8) & (G ^@y8 z)) supposing { (F ^x5) }> from the inference-
queue.
```

```
                                        # 5
                                        interest: ((G ^x5 ^@y8) & (G ^@y8 ^@y9))    supposition: { (F ^x5) }
                                        For interest 4 by EG
---------------------------------------------------------------------------
Retrieving #<Interest 5: ((G ^x5 ^@y8) & (G ^@y8 ^@y9)) supposing { (F ^x5) }> from the inference-queue.
                                        # 6
                                        interest: (G ^@y8 ^@y9)    supposition: { (F ^x5) }
                                        For interest 5 by adjunction
                                        Conclusion #7 discharges interest #6
                                        # 7
                                        interest: (G ^x5 ^@y8)    supposition: { (F ^x5) }
                                        For interest 5 by adjunction using conclusion 7
                                        Conclusion #7 discharges interest #7
---------------------------------------------------------------------------
Retrieving #<Interest 7: (G ^x5 ^@y8) supposing { (F ^x5) }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Interest 6: (G ^@y8 ^@y9) supposing { (F ^x5) }> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 9> from the inference-queue.

  # 10   (F @y7)    supposition: { (F ^x5) }
  Inferred by support-link #8 from { 9 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 11   (G @y6 @y7)    supposition: { (F ^x5) }
  Inferred by support-link #9 from { 9 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 12   ((G ^x5 @y6) & (G @y6 @y7))    supposition: { (F ^x5) }
  Inferred by support-link #10 from { 11 , 7 } by adjunction
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 5

  # 13   (some z)((G ^x5 @y6) & (G @y6 z))    supposition: { (F ^x5) }
  Inferred by support-link #11 from { 12 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 4

  # 14   (some y)(some z)((G ^x5 y) & (G y z))    supposition: { (F ^x5) }
  Inferred by support-link #12 from { 13 } by EG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 3

  # 15   ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z)))
  Inferred by support-link #13 from { 14 } by conditionalization
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 2

  # 16   (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))
  Inferred by support-link #14 from { 15 } by UG
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
  This node discharges interest 1
             ---------------------------------------
             #<Node 16> answers #<Query #1: (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))>
             ---------------------------------------
           ALL QUERIES HAVE BEEN ANSWERED DEDUCTIVELY.

================= ULTIMATE EPISTEMIC INTERESTS ==================
  Interest in (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))
  is answered affirmatively by conclusion 16
```

```
===============================================================================
ARGUMENT #1
This is a deductive argument for:
      (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))
 which is of ultimate interest.

 1.  (all x)((F x) -> (some y)((F y) & (G x y)))     given
 2.  ((F x0) -> (some y)((F y) & (G x0 y)))     UI from { 1 }
        |-----------------------------------------------------------
        | Suppose:  { (F ^x5) }
        |-----------------------------------------------------------
        | 3.  (F ^x5)     supposition
        | 4.  (some y)((F y) & (G ^x5 y))     modus-ponens2 from { 2 , 3 }
        | 5.  ((F @y6) & (G ^x5 @y6))     EI from { 4 }
        | 7.  (G ^x5 @y6)     simp from { 5 }
        | 6.  (F @y6)     simp from { 5 }
        | 8.  (some y)((F y) & (G @y6 y))     modus-ponens2 from { 2 , 6 }
        | 9.  ((F @y7) & (G @y6 @y7))     EI from { 8 }
        | 11.  (G @y6 @y7)     simp from { 9 }
        | 12.  ((G ^x5 @y6) & (G @y6 @y7))     adjunction from { 11 , 7 }
        | 13.  (some z)((G ^x5 @y6) & (G @y6 z))     EG from { 12 }
        | 14.  (some y)(some z)((G ^x5 y) & (G y z))     EG from { 13 }
 15.  ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z)))     conditionalization from { 14 }
 16.  (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))     UG from { 15 }
===============================================================================
```

If we attempt to do this same problem without using the EI discount, we get infinite cycling (of length 4) beginning with conclusion 9.

```
Given premises:
     (all x)((F x) -> (some y)((F y) & (G x y)))    justification = 1
Ultimate epistemic interests:
     (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))    interest = 1


  # 1   (all x)((F x) -> (some y)((F y) & (G x y)))
  given
  maximal-degree-of-support: 1

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

  # 2   ((F x0) -> (some y)((F y) & (G x0 y)))
  Inferred by support-link #1 from { 1 } by UI
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 2> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Query #1: (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))> from the inference-queue.
                                        # 1
                                        interest: (all x)((F x) -> (some y)(some z)((G x y) & (G y z)))
                                        This is of ultimate interest
                                        # 2
                                        interest: ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z)))
                                        For interest 1 by UG
---------------------------------------------------------------------------
Retrieving #<Interest 2: ((F ^x5) -> (some y)(some z)((G ^x5 y) & (G y z))) supposing {  }> from the
inference-queue.

  # 3   (F ^x5)    supposition: { (F ^x5) }
  supposition
  maximal-degree-of-support: 1

                                        # 3
                                        interest: (some y)(some z)((G ^x5 y) & (G y z))    supposition: { (F ^x5) }
                                        For interest 2 by conditionalization
---------------------------------------------------------------------------
Retrieving #<Node 3> from the inference-queue.

  # 4   (some y)((F y) & (G ^x5 y))    supposition: { (F ^x5) }
  Inferred by support-link #2 from { 2 , 3 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 4> from the inference-queue.
```

# 5   ((F @y6) & (G ^x5 @y6))   supposition: { (F ^x5) }
Inferred by support-link #3 from { 4 } by EI

```
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 5> from the inference-queue.

  # 6   (F @y6)    supposition: { (F ^x5) }
  Inferred by support-link #4 from { 5 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 7   (G ^x5 @y6)    supposition: { (F ^x5) }
  Inferred by support-link #5 from { 5 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 6> from the inference-queue.

  # 8   (some y)((F y) & (G @y6 y))    supposition: { (F ^x5) }
  Inferred by support-link #6 from { 2 , 6 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 7> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 8> from the inference-queue.
```

# 9   ((F @y7) & (G @y6 @y7))   supposition: { (F ^x5) }
Inferred by support-link #7 from { 8 } by EI

```
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 9> from the inference-queue.

  # 10   (F @y7)    supposition: { (F ^x5) }
  Inferred by support-link #8 from { 9 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.

  # 11   (G @y6 @y7)    supposition: { (F ^x5) }
  Inferred by support-link #9 from { 9 } by simp
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 10> from the inference-queue.

  # 12   (some y)((F y) & (G @y7 y))    supposition: { (F ^x5) }
  Inferred by support-link #10 from { 2 , 10 } by modus-ponens2
  maximal-degree-of-support: 1
  This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 11> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 12> from the inference-queue.
```

```
 # 13   ((F @y8) & (G @y7 @y8))    supposition: { (F ^x5) }
 Inferred by support-link #11 from { 12 } by EI
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 13> from the inference-queue.

 # 14   (F @y8)    supposition: { (F ^x5) }
 Inferred by support-link #12 from { 13 } by simp
 maximal-degree-of-support: 1
 This node encodes a deductive argument.

 # 15   (G @y7 @y8)    supposition: { (F ^x5) }
 Inferred by support-link #13 from { 13 } by simp
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 14> from the inference-queue.

 # 16   (some y)((F y) & (G @y8 y))    supposition: { (F ^x5) }
 Inferred by support-link #14 from { 2 , 14 } by modus-ponens2
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
---------------------------------------------------------------------------
Retrieving #<Node 15> from the inference-queue.
---------------------------------------------------------------------------
Retrieving #<Node 16> from the inference-queue.

 # 17   ((F @y9) & (G @y8 @y9))    supposition: { (F ^x5) }
 Inferred by support-link #15 from { 16 } by EI
 maximal-degree-of-support: 1
 This node encodes a deductive argument.
---------------------------------------------------------------------------
                                                                           .
                                                                           .
                                                                           .
```

All of this reasoning is legitimate, and might be required for another problem, so it cannot be blocked, but only delayed. That is the effect of lowering the priority.

A slot has been added to interests for *discharge-condition*. This is a condition that a conclusion must satisfy in order to discharge the interest. Its use here is in the rule UG.

UG *sequent interest*

If the interest-formula *p* of *interest* isa universal generalization:

- Let *e-vars* be the interest-variables of interest, let *x* be the variable bound by the initial universal quantifier, and let *p** be the matrix of *p* (the formula immediately following the initial quantifier).
  - If *e-vars* is empty, let *fun* be a new variable ^@*y*. Let *term* be *fun*. If *e-vars* is nonempty, let *fun* be a new skolem-function ^@*y*, and let *term* be constructed by appending *fun* to the front of the list of *e-vars*.
  - Let *p*** be the result of substituting *term* for *x* in *p**.
  - Let *ug-condition* be the condition (applied to variables *node* and *unifier*) that *fun* does not occur in the result of applying the first member of *unifier* to the node-supposition of *node* and *fun* does not occur in the second member of *unifier*.
  - Where *sup* is the interest-supposition of *interest*, adopt interest in ⟨*sup*,*p***⟩ with target-interest *interest*, and let the discharge-condition of the new interest be *ug-condition*.

The ug-condition implements the standard constraint on the use of UG requiring that in order to infer $(\forall x)(F\ x)/Y$ from $(F\ z)/Y$, z must not occur in Y.

The final miscellaneous detail concerns the complexity computation used in ordering the *inference-queue*. The ordering proceeds in terms of *sequent-complexity*, which is defined in terms of *formula-complexity*, which in turn is defined in terms of *complexity*. The latter simply counts the number of atoms or strings occurring in an expression, with two exceptions. The first exception is formulated in terms of a constant called the **skolem-multiplier**. This is used in computing the complexity of a formula, and has the effect of multiplicatively increasing the complexity of any term beginning with a skolem-function by a factor equal to the value of the **skolem-multiplier**. This has the effect of lowering the priority of reasoning with formulas containing skolem-terms that contain many layers of simpler skolem-terms, e.g., (*s* (*s* (*s* (*s x*)))) The default value of **skolem-multiplier** is 10, which was chosen experimentally as giving the most efficient performance on the test problems. The second exception concerns quantified formulas. Our previous definition of complexity makes a quantified formula more complex than its instances. But it is generally better to give quantified formulas a higher priority than their instances. So in computing the complexity of a quantified formula, we ignore quantifiers and we discount the complexity of the matrix by a factor *quantifier-discount*, whose default value is .95.

The formula-complexity of a formula is its complexity unless the formula is the node-formula of a supposition that does not contain any skolem-constants or skolem-functions. The latter formulas are stored in the list *skolem-free-suppositions*, and their formula-complexity is taken to be 0. If a sequent has an empty supposition, its sequent-complexity is either 1 or its formula-complexity, whichever is larger. If the sequent-supposition is nonempty, the sequent-complexity is the sum of the latter number and all the complexities of members of the supposition that are not in *skolem-free-suppositions*. It is to be emphasized that this definition of complexity is entirely ad hoc, and was arrived at experimentally rather than being theoretically motivated. Users are encouraged to experiment with different definitions of SEQUENT-COMPLEXITY and FORMULA-COMPLEXITY.

3. Defeasible Reasoning with Free Variables

The preceding has concerned the use of skolemization and unification in deductive reasoning. Skolemized formulas can also be employed in defeasible reasoning, and unless explicit exceptions are incorporated into OSCAR's reasoning, defeasible reasons will be applied using unification, just as for deductive inference rules. Is that legitimate? Consider a simple case. Suppose 'Fx' is a conclusive reason for 'Hx', and 'Hx' defeasible reason for 'Gx'. If the reasoner is interested in establishing (∀x)(Fx ⊃ Gx), a supposition Fx, with free 'x', will be made, and an interest in Gx/{Fx} will be adopted. A conclusion Hx/{Fx} will be drawn, and Hx is a defeasible reason for Gx. So from Hx/{Fx}, a defeasible inference will be made to Gx/{Fx}. That satisfies the interest, and so (∀x)(Fx ⊃ Gx) will be inferred. It seems clear that this is a legitimate form of inference. If for any choice of 'a', 'Fa' is a reason for 'Ga', then we ought to be able to infer defeasibly that everything that is F is G. However, it must be recognized that this constitutes a significant extension of the use of defeasible reasons in reasoning with closed formulas.

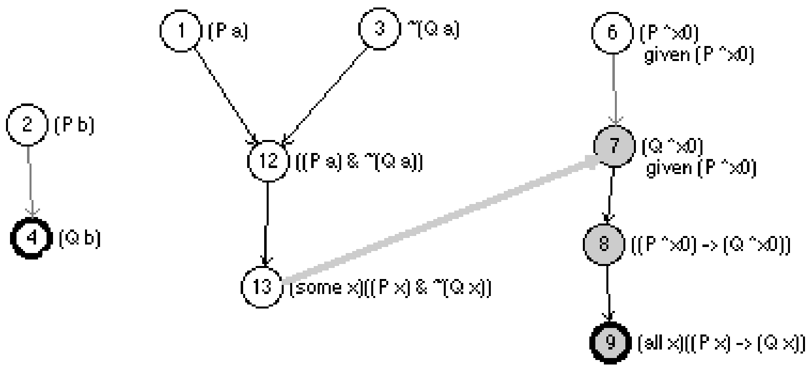
Whenever a defeasible inference is made, the reasoner must adopt interest in the rebutting and undercutting defeaters. If defeasible reasoning with free variables were a simple extension of defeasible reasoning with closed formulas, then having inferred Gx/{Fx} from Hx/{Fx}, the reasoner would adopt interest in ~Gx/{Fx}. However, if the reasoner could get that conclusion, it could go on to infer (∀x)(Fx ⊃ ~Gx). That is too strong a requirement for defeat. It should suffice that the reasoner can infer (∃x)(Fx & ~Gx). The undercutting defeater should similarly be (∃x)(Fx & (Hx @ Gx)). More generally, if 'Hxy' is a defeasible reason for 'Gxy' and the reasoner infers Gax/{Fax & P} from Hax/{Fax & P}, where 'x' does not occur free in P, then the undercutting defeater will be (∃x)(Fax & (Hax @ Gax))/{P} and the rebutting defeater will be (∃x)(Fax & ~Gax)/{P}.

To illustrate, consider the following simple problem:

```
Given premises:
   (P a)    justification = 1.0
```

```
    (P b)    justification = 1.0
    ~(Q a)    justification = 1.0
Ultimate epistemic interests:
    (Q b)    interest = 0.9
    (all x)((P x) -> (Q x))    interest = 0.9

   FORWARDS PRIMA FACIE REASONS
     pf-reason_1.1:   {(P x)} ||=> (Q x)  variables = {x}   strength = 0.9
```



```
  # 1
  (P a)
given
  Inferred by:
              support-link #1 from {  } by given
  This discharges interest 13
  # 2
  (P b)
given
  Inferred by:
              support-link #2 from {  } by given
  # 3
  ~(Q a)
given
  Inferred by:
              support-link #3 from {  } by given
  defeatees: { link 5 for node 5 }
  This discharges interest 12
                                   # 1
                                   interest: (all x)((P x) -> (Q x))
                                   This is of ultimate interest
                                   # 2
                                   interest: ((P ^x0) -> (Q ^x0))
                                   For interest 1 by UG
                                   This interest is discharged by node 8
                                   # 3
                                   interest: (Q b)
                                   This is of ultimate interest
  # 4
  (Q b)
  Inferred by:
              support-link #4 from { 2 } by pf-reason_1.1
  This discharges interest 3
             ==========================================
             Justified belief in (Q b)
             answers #<Query #1: (Q b)>
             ==========================================
  # 6
  (P ^x0)    supposition: { (P ^x0) }
supposition
```

```
generated by interest 2
                                   # 8
                                   interest: (Q ^x0)    supposition: { (P ^x0) }
                                   For interest 2 by conditionalization
                                   This interest is discharged by node 7
 # 7
 (Q ^x0)    supposition: { (P ^x0) }                  DEFEATED
 Inferred by:
             support-link #6 from { 6 } by pf-reason_1.1  defeaters: { 13 }   DEFEATED
 This discharges interest 8
                                   # 10
                                   interest: (some x)((P x) & ~(Q x))
                                   Of interest as a defeater for support-link 6 for node 7
                                   This interest is discharged by node 13
 # 8
 ((P ^x0) -> (Q ^x0))                 DEFEATED
 Inferred by:
             support-link #7 from { 7 } by conditionalization   DEFEATED
 This node is inferred by discharging interest #2
 # 9
 (all x)((P x) -> (Q x))                  DEFEATED
 Inferred by:
             support-link #8 from { 8 } by UG   DEFEATED
 This node is inferred by discharging interest #1
            ========================================
            Justified belief in (all x)((P x) -> (Q x))
            answers #<Query #2: (all x)((P x) -> (Q x))>
            ========================================
 # 12
 ((P a) & ~(Q a))
 Inferred by:
             support-link #13 from { 1 , 3 } by adjunction
 This node is inferred by discharging interest #11
 # 13
 (some x)((P x) & ~(Q x))
 Inferred by:
             support-link #14 from { 12 } by EG
 defeatees: { link 6 for node 7 }
 This node is inferred by discharging interest #10
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            #<Node 7> has become defeated.
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            #<Node 8> has become defeated.
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            #<Node 9> has become defeated.
            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
            ========================================
            Lowering the undefeated-degree-of-support of (all x)((P x) -> (Q x))
            retracts the previous answer to #<Query #2: (all x)((P x) -> (Q x))>
            ========================================
============================================================
```

A similar complication concerns INVERT-CONTRADICTIONS. As it stands, that rule has the reasoner infer the negations of nearest-defeasible-ancestors. However, those may contain free variables. It must accordingly introduce existential quantifiers, in the same manner as is done with defeaters for inferences using free variables.