Natural Deduction

John L. Pollock
Department of Philosophy
University of Arizona
Tucson, Arizona 85721
(e-mail: pollock@arizona.edu)

Most automated theorem provers are clausal-form provers based on variants of resolution-refutation. In my [1990], I described the theorem prover OSCAR that was based instead on natural deduction. Some limited evidence was given suggesting that OSCAR was suprisingly efficient. The evidence consisted of a handful of problems for which published data was available describing the performance of other theorem provers. This evidence was suggestive, but based upon too meager a comparison to be conclusive. The question remained, "How does natural deduction compare with resolution-refutation?" In the ensuing seven years, OSCAR has evolved in important ways, and other developments have made it possible to collect more accurate comparative data. Specifically, the creation of the TPTP library of problems for theorem provers,[1] and the availability of important theorem provers on the world wide web, make objective comparisons easier. These developments recently inspired Geoff Sutcliffe, one of the founders of the TPTP library, to issue a challenge to OSCAR. At CADE-13, a competition was held for clausal-form theorem provers.[2] Otter was one of the most successful contestants. In addition, Otter is able to handle problems stated in natural form (as opposed to clausal form), and Otter is readily available for different platforms.[3] Sutcliffe selected 212 problems from the TPTP library, and suggested that OSCAR and Otter run these problems on the same hardware. This "Shootout at the ATP corral" took place, with the result that OSCAR was on the average 40 times faster than Otter. In addition, OSCAR was able to find proofs for 16 problems on which Otter failed, and Otter was able to find proofs for 3 problems on which OSCAR failed. Taking into account that Otter was written in C and OSCAR in LISP, the speed difference of the algorithms themselves could be as much as an order of magnitude greater. Apparently, natural deduction has some advantages over resolution-refutation.[4]

Most researchers in automated theorem proving are relatively unfamiliar with natural deduction, and there has been little work on the foundations of automated natural deduction theorem proving. This paper presents a theoretical analysis of natural deduction. First, a brief description will be given of a natural deduction system for sentential logic (i.e., no quantifiers or variables). Then it will be shown how to extend it to first-order logic using skolemization and unification.

This work was supported in part by NSF grant no. IRI-9634106.

[1] Suttcliffe, Suttner, and Yemenis [1994].

[2] The results are given in Sutcliffe [1996].

[3] Otter can be downloaded from http://www.mcs.anl.gov/home/mccune/ar/otter/index.html.

[4] A detailed report of the shootout are being prepared by Pollock and Sutcliffe. In the meantime, a preliminary version of the report can be downloaded from http://www.u.arizona.edu/~pollock.

1. Natural Deduction

I take two characteristics to be definitive of natural deduction: (1) the reasoning proceeds by bidirectional search; (2) what are manipulated are sequents rather than formulas. I have elsewhere[5] described natural deduction systems as performing "interest-driven suppositional reasoning", which comes to the same thing. The sense in which it is *suppositional* reasoning is that it reasons with sequents rather than formulas. A sequent is a pair consisting of a formula and a set of formulas (the supposition of the sequent). Sequents will be written in the form p/X, where p is a formula and X is a set of formulas. Semantically, I will assign truth values to sequents, treating p/X as equivalent to the corresponding conditional $\Pi X \supset p$ (where ΠX is the conjunction of members of X). A sequent with an empty supposition will be identified with its sequent-formula.

In an interest-driven reasoner, we maintain two lists of sequents—conclusions and interests. Initially, the list of conclusions contains just the given premises, and the list of interests contains just the sequents we are attempting to derive (the *ultimate-interests*). Interest-driven reasoning proceeds by bidirectional search, reasoning forwards from the conclusions and backwards from the interests until the two strands of reasoning are brought together. Reasoning from conclusions to conclusions is performed in accordance with *forwards-inference-rules* and reasoning from interests to interests is performed in accordance with *backwards-inference-rules*. These will typically be disjoint sets of inference rules. (For further discussion of this aspect of interest-driven reasoning, see my [1990] or [1995].) In sentential logic, the rules are applied by pattern matching. Here is a simple example of an interest-driven suppositional reasoner in action:

Given premises: ----
Ultimate epistemic interests:
  ((p ⊃ q) ∨ (q ⊃ p))

                    # 1
                    interest: ((p ⊃ q) ∨ (q ⊃ p))
                    This is of ultimate interest
                    # 3
                    interest: (~(q ⊃ p) ⊃ (p ⊃ q))
                    For interest 1 by DISJUNCTION CONVERSION
                    This interest is discharged by node 6
 # 1
 ~(q ⊃ p) / { ~(q ⊃ p) }
supposition
 generated by interest 3
                    # 4
                    interest: (p ⊃ q) / { ~(q ⊃ p) }
                    For interest 3 by CONDITIONALIZATION
                    This interest is discharged by node 5
 # 2
 q / { ~(q ⊃ p) }

[5] My [1990] and [1995].

from { 1 } by NEG-CONDIT

5

interest: q / { p , ~(q ⊃ p) }
For interest 4 by CONDITIONALIZATION
This interest is discharged by node 2

5

(p ⊃ q) / { ~(q ⊃ p) }
from { 2 } by CONDITIONALIZATION
This node is inferred by discharging interest #4

6

(~(q ⊃ p) ⊃ (p ⊃ q))
from { 5 } by CONDITIONALIZATION
This node is inferred by discharging interest #3

7

((p ⊃ q) ∨ (q ⊃ p))
from { 6 } by DISJUNCTION CONVERSION
This node is inferred by discharging interest #1

A "simple" forwards-inference-rule will be written in the form $\{f_1,...,f_n\} \Rightarrow f$, where the f_i are formulas containing schematic variables (I will omit the braces when there is a single premise). An example would be $\ulcorner\{p, (p \supset q)\} \Rightarrow q\urcorner$, where p and q are schematic variables. Forwards reasoning proceeds by finding pattern-matches of conclusion-formulas with the premises of forwards-rules. Simple backwards-inference-rules are written similarly, but in addition, backwards-inference-rules may have *discharges*. For example, CONDITIONALIZATION can be written as follows:

$$q \Leftarrow (p \supset q), \text{ with discharge } p.$$

This means that given an interest in $(p \supset q)/X$, the reasoner makes a new supposition p, recorded as a conclusion whose sequent is $p/\{p\}$, and adopts interest in $q/X\cup\{p\}$. If a conclusion q/Y is subsequently obtained where $Y \subseteq X\cup\{p\}$, the reasoner will infer $(p \supset q)/X$, dropping ("discharging") the supposition p. This is illustrated in conclusions 1 and 6 above.

Backwards reasoning creates new interests. An *interest-link* records the connection between an interest (the *target-interest* of the link) and the set of new interests derived from it (the *link-interests*). For example, using the backwards-inference-rule ADJUNCTION, interest in (p & q) will generate interest in p and interest in q. The latter two interests comprise the link-interests of an interest-link whose target is the interest in (p & q). An interest-link is said to be a *right-link* of its link-interests. A conclusion *discharges* an interest iff (1) the conclusion-formula is the same as the interest-formula, and (2) the conclusion-supposition is a subset of the interest-supposition. When all of the link-interests are discharged, we say that the link itself is discharged, and the reasoner infers the link-target, adding it to the set of conclusions.

Most inference-rules of use in logic are simple backwards or forwards-inference-rules, but not all inference-rules can be cast in that form. Consider the following rule of REDUCTIO:

> Given an interest in p/X, suppose $\sim p$. Then for every conclusion q/Y where $Y \subseteq X\cup\{\sim p\}$, adopt interest in $\sim q/X\cup\{\sim p\}$. If the latter is concluded, infer p/X.

This is a backwards-inference-rule, but it must be regarded as having two different kinds of premises. The conclusion is p, and the premises are $q / \{\sim p\}$ and $\sim q / \{\sim p\}$. However, the inference-rule does not instruct the reasoner that when interest is adopted in p, then interest should be adopted in both $q / \{\sim p\}$ and $\sim q / \{\sim p\}$. Rather, the instruction is that if $q / \{\sim p\}$ has already been concluded, then interest should be adopted in $\sim q / \{\sim p\}$. We can express this by saying that REDUCTIO has two premises—one a *forwards-premise* and the other a *backwards-premise*. In deploying a backwards-inference-rule, the reasoner does not adopt interest in the forwards-premises. Only the backwards-premises give rise to interests, and they only do so when conclusions have already been drawn that instantiate the forwards-premises.

We can distinguish between three different kinds of backwards-inference-rules. *Simple backwards-inference-rules* have only backwards-premises—no forwards premises. In formal logic, most backwards-inference-rules are simple backwards-reasons, but my experience has been that in defeasible reasoning there are few simple backwards-inference-rules. *Mixed backwards-inference-rules* are backwards-inference-rules having both forwards- and backwards-premises. There can also be *degenerate backwards-inference-rules* that have only forwards-premises. Given an interest in the conclusion of a degenerate backwards-inference-rules, if the reasoner draws conclusions instantiating the forwards-premises, then a conclusion will be drawn instantiating the conclusion of the rule, but no new interests will be adopted.[6]

To accommodate both forwards and backwards premises in backwards-inference-rules, they should now be written more generally in the form:

$\{f_1,...,f_n\}\{b_1,...,b_m\} \Leftarrow c$ [with discharge p]

where $f_1,...,f_n$ are the forwards-premises and $b_1,...,b_m$ are the backwards-premises. For example, we can write the above rule of reductio-ad-absurdum as follows:

$\{q\}\{\sim q\} \Leftarrow p$ with discharge $\sim p$.

I will continue to write simple backwards-inference-rules in the form $\ulcorner\{b_1,...,b_m\} \Leftarrow c\urcorner$ rather than $\ulcorner\{\}\{b_1,...,b_m\} \Leftarrow c\urcorner$.

Forwards-inference-rules can also have both backwards- and forwards-premises. That is very useful in defeasible reasoning,[7] but I have not found it useful in formal logic, so that complication will be ignored in the present paper.

Backwards reasoning with a mixed or degenerate backwards-inference-rule will create an *interest-scheme* rather than an interest-link. An interest-scheme uses the target-interest to partially instantiate the forwards-premises. Those partially instantiated premises are recorded in the interest-scheme, and when conclusions are inferred that fully instantiate them, the resulting instantiation is used to instantiate the backwards-premises if there are any, or in the case of degenerate backwards-inference-rules to instantiate the conclusion of the rule. Once the backwards-premises are instantiated, interest is adopted in them and an interest-link is constructed to link them to the target interest. To illustrate, consider a backwards-inference-rule $\ulcorner\{Fxy\}\{Gyz\} \Leftarrow Hxz\urcorner$ where x,y,z are schematic variables. Given an interest in Hab, the forwards-premise is partially instantiated to $\ulcorner Fay\urcorner$, and an interest-scheme is constructed recording the instance $\ulcorner\{Fay\}\{Gyb\} \Leftarrow Hab\urcorner$ of the inference-rule. If $\ulcorner Fac\urcorner$ is concluded, that is used to fully instantiate the interest-scheme to produce the instance $\ulcorner\{Fac\}\{Gcb\} \Leftarrow Hab\urcorner$. This is recorded as an interest-link

[6] See my [1996] for a discussion of the usefulness of such rules in defeasible reasoning.

[7] My [1996].

and interest is adopted in ⌜Gcb⌝. If ⌜Gcb⌝ is concluded, then ⌜Hab⌝ will be inferred. The difference between interest-schemes and interest-links is that the former have forwards-premises that have not yet been instantiated by conclusions.

We can regard a natural deduction reasoner as consisting of a simple loop. As new conclusions and interests are created, they are placed in a list called the *inference-queue*, which can be prioritized however we like, and then the elements of the inference-queue are retrieved one at a time:

loop
- Choose a member q of the inference-queue having maximal priority.
- Delete q from the inference-queue.
- If q is an interest, REASON-BACKWARDS-FROM q.
- If q is a conclusion, REASON-FORWARDS-FROM q, DISCHARGE-INTEREST-SCHEMES-FROM q, and DISCHARGE-INTERESTS-FROM q.
- If all ultimate-interests are discharged or the inference-queue is empty, return from the loop.

REASON-BACKWARDS-FROM *interest*

If the interest-sequent of *interest* is p/X, then given any backwards-inference-rule $\{f_1,...,f_n\}\{b_1,...,b_m\} \Leftarrow c$ with discharge d, if there is a pattern-match m matching c with p:
- if there are no forwards-premises,
  - if d is non-NIL:
    - add $d/\{d\}$ to the list of conclusions and put it on the inference-queue.
    - construct interests in the sequents $\mathrm{m}(f_i)/X \cup \{\mathrm{m}(d)\}$ and construct an interest-link whose resultant-interest is *interest* and whose link-interests consist of the new interests;
  - if d is NIL, construct interests in the sequents $\mathrm{m}(f_i)/X$ and construct an interest-link whose resultant-interest is *interest* and whose link-interests consist of the new interests;
  - if some list of conclusions discharges the interest-link, infer *interest* from those conclusions, adding it to the list of conclusions and placing it in the inference-queue.
- if there are forwards-premises,
  - if d is non-NIL:
    - add $d/\{d\}$ to the list of conclusions and put it on the inference-queue.
    - construct an interest-scheme $\{\mathrm{m}(f_1),...,\mathrm{m}(f_n)\}\{\mathrm{m}(b_1),...,\mathrm{m}(b_m)\} \Leftarrow p$, with supposition $X \cup \{\mathrm{m}(d)\}$ and target-interest *interest*;
  - if d is NIL, construct an interest-scheme $\{\mathrm{m}(f_1),...,\mathrm{m}(f_n)\}\{\mathrm{m}(b_1),...,\mathrm{m}(b_m)\} \Leftarrow p$, with supposition X and target-interest *interest*;
  - if a list of conclusions $\mathrm{m}(f_1)/Y_1,...,\mathrm{m}(f_n)/Y_n$ have already been drawn where the $Y_i \subseteq X$:
    - if there are backwards-premises, adopt interest in $\mathrm{m}(b_1),...,\mathrm{m}(b_m)$ and construct an interest-link linking those interests to *interest* and having the discharge p. If conclusions $\mathrm{m}(b_1)/Z_1,...,\mathrm{m}(b_m)/Z_m$ have already been drawn where the $Z_i \subseteq X$, infer *interest*;
    - if there are no backwards-premises, infer *interest*.

REASON-FORWARDS-FROM *conclusion*

Given any forwards-inference-rule $\{f_1,...,f_n\} \Rightarrow f$ and any list $p_1/Y_1,...,p_n/Y_n$ of conclusion-sequents (possibly with repetition) that includes the conclusion-sequent of *conclusion*, if there is a pattern-match m matching the list $f_1,...,f_n$ with the list $p_1,...,p_n$, infer

$m(f)/(m(Y_1) \cup ... \cup m(Y_n))$ from the sequents $p_1/Y_1,...,p_n/Y_n$, adding the former to the list of conclusions and placing it in the inference-queue.

DISCHARGE-INTEREST-SCHEMES-FROM *conclusion*

Given any interst-scheme $\{f_1,...,f_n\}\{b_1,...,b_m\} \Leftarrow c$ with discharge p and supposition X, and any list $p_1/Y_1,...,p_n/Y_n$ of conclusion-sequents (possibly with repetition) that includes the conclusion-sequent of *conclusion*, if each $Y_i \subseteq X$ and there is a pattern-match m matching the list $f_1,...,f_n$ with the list $p_1,...,p_n$:

- if there are backwards-premises, adopt interest in $m(b_1),...,m(b_m)$ and construct an interest-link linking those interests to the target interest of the interest-scheme and having the discharge p. If conclusions $m(b_1)/Z_1,...,m(b_m)/Z_m$ have already been drawn where the $Z_i \subseteq X$, infer the target-interest;
- if there are no backwards-premises, infer the target-interest.

DISCHARGE-INTERESTS-FROM *conclusion*

If *conclusion* discharges *interest*, then for every interest-link whose link-interests contain *interest*, if *conclusion* together with some other conclusions jointly discharge the link, infer the link-target from those conclusions, adding it to the list of conclusions and placing it in the inference-queue.

A natural deduction system for sentential logic will be sound and complete if contains an appropriate selection of inference-rules. In choosing inference rules, the basic desideratum is that they allow complex formulas to be broken into simpler parts. Let us define $\ulcorner \neg p \urcorner$ to be q if $p = \ulcorner {\sim} q \urcorner$, and $\ulcorner \neg p \urcorner$ is $\ulcorner {\sim} p \urcorner$ otherwise. Then the following set of backwards- and forwards-rules is sound and complete.

<u>Forwards-Inference-Rules</u>

SIMPLIFICATION

$(p \,\&\, q) \Rightarrow p$

$(p \,\&\, q) \Rightarrow q$

BICONDITIONAL ELIMINATION

$(p \equiv q) \Rightarrow (p \supset q),(q \supset p)$

NEGATION ELIMINATION

${\sim}{\sim}p \Rightarrow p$

MODUS PONENS

$\{p,(p \supset q)\} \Rightarrow q$

MODUS TOLLENS

$\{{\sim}q,(p \supset q)\} \Rightarrow {\sim}p$

DISJUNCTION CONVERSION

$(p \vee q) \Rightarrow (\neg p \supset q)$

DE MORGAN

${\sim}(p \,\&\, q) \Rightarrow (\neg p \vee \neg p)$

DISJUNCTION NEGATION

${\sim}(p \vee q) \Rightarrow (\neg p \,\&\, \neg p)$

<u>Backwards-Inference-Rules</u>

ADJUNCTION

$\{p,q\} \Leftarrow (p \,\&\, q)$

BICONDITIONAL INTRODUCTION

$(p \supset q),(q \supset p) \Leftarrow (p \equiv q)$

NEGATION INTRODUCTION

$p \Leftarrow {\sim}{\sim}p$

CONDITIONALIZATION

$q \Leftarrow (p \supset q)$, with discharge p

DISJUNCTION CONVERSION

$(\neg p \supset q) \Leftarrow (p \vee q)$

$(\neg q \supset p) \Leftarrow (p \vee q)$

DE MORGAN

$(\neg p \vee \neg q) \Leftarrow {\sim}(p \,\&\, q)$

DISJUNCTION NEGATION

$\{\neg p \,,\, \neg q\} \Leftarrow {\sim}(p \vee q)$

CONDITIONAL NEGATION

$\sim(p \supset q) \Rightarrow (p \;\&\; \neg p)$

CONDITIONAL NEGATION

$\{p\ ,\ \neg q\} \Leftarrow \sim(p \supset q)$

REDUCTIO

$\{q\}\{\sim q\} \Leftarrow p$, with discharge $\sim p$

FORTUITOUS REDUCTIO

$\{q\ ,\ \sim q\}\{\} \Leftarrow p$

In the interest of efficiency, restrictions can be placed on the rule REDUCTIO requiring that *p* be a literal and *q* be either a literal or a conditional. This prevents the reasoner from following different backwards search paths to the same interests. Similarly, in FORTUITOUS-REDUCTIO, *p* can be required to be a literal and *q* to be atomic. The correctness of these restrictions will be proven in the next section.

2. Completeness

A subset of the rules of the previous section can be shown to be complete. This subset deletes MODUS PONENS, MODUS TOLLENS, and the second DISJUNCTION CONVERSION rule from the preceding list.

Let us define an *ND-problem* to be a pair $\langle\Gamma,p\rangle$ where Γ (the *conclusions*) is a set of formulas and *p* (the *interest*) is a formula. An ND-problem $\langle\Gamma,p\rangle$ is *valid* iff $\Gamma \vDash p$ (where '$\vDash$' is the semantical implication relation). An ND-problem $\langle\Gamma,p\rangle$ is *solved* iff either $p\in\Gamma$ or there is a *q* such that $q\in\Gamma$ and $\ulcorner\sim q\urcorner\in\Gamma$. A *reduction-rule* is a relation whose extension is a set of pairs $\langle\rho,\Pi\rangle$ where ρ is an ND-problem and Π is a finite set of ND-problems. A reduction-rule is *valid* iff for every $\langle\rho,\Pi\rangle$ in its extension, ρ is valid if all members of Π are valid. A reduction-rule is *sound* iff it is valid and for every $\langle\rho,\Pi\rangle$ in its extension, if ρ is valid then all members of Π are valid. A *solution* to an ND-problem (relative to a set of reduction rules) is a finite (upside down) tree of reductions (a "reduction-tree"), where the top node is the problem solved and the leaf nodes are solved ND-problems. A set of reduction-rules is *complete* iff every valid ND-problem has a solution.

The rules for natural-deduction reasoning can be regarded as reduction-rules for ND-problems. The backwards-inference-rules reduce ND-problems to problems with different interests (and sometimes different conclusions). For example, ADJUNCTION reduces a problem of the form $\langle\Gamma,(p\&q)\rangle$ to the pair of problems $\langle\Gamma,p\rangle$ and $\langle\Gamma,q\rangle$. CONDITIONALIZATION reduces a problem of the form $\langle\Gamma,(p \supset q)\rangle$ to the problem $\langle\Gamma\cup\{p\},q\rangle$. REDUCTIO reduces a problem of the form $\langle\Gamma,p\rangle$ (where *p* is a literal) to a problem of the form $\langle\Gamma\cup\{\neg p\},\neg q\rangle$ where $q\in\Gamma$ and *q* is either a conditional or a literal. Note that REDUCTIO is taken to reduce a problem to any single problem produced in this way rather than to the set of problems produced in this way. It will follow from the discussion below that it makes no difference to completeness which reducing problem is selected.

In actual practice, the forwards-inference-rules are used to add formulas to the set of conclusions, but for present purposes I will construe them more narrowly as rewrite rules. A rewrite rule is a reduction-rule that reduces an ND-problem $\langle\Gamma\cup\{p\},q\rangle$ to a set of ND-problems $\langle\Gamma^*,r\rangle$ where $p\notin\Gamma^*$. For example, SIMPLIFICATION will be taken to reduce a problem of the form $\langle\Gamma\cup\{(p\&q)\},r\rangle$ to the problem $\langle\Gamma\cup\{p,q\},r\rangle$. In the latter, $\ulcorner(p\&q)\urcorner$ has been removed from the set of conclusions. If a set of rules is complete when the forwards-inference-rules are construed as rewrite rules, then a fortiori, it is complete if they are construed more generally as rules that simply add formulas to the list of conclusions. Note that MODUS PONENS and MODUS TOLLENS cannot be construed as rewrite rules, so it is important for the present completeness proof that these rules are not needed. These rules can, however, be cast as valid reduction rules. For example MODUS PONENS can be taken to reduce $\langle\Gamma\cup\{p,(p\supset q)\},r\rangle$ to $\langle\Gamma\cup\{p,(p\supset q),q\},r\rangle$.

Valid reduction-rules correspond to valid inference-rules. Sound reduction-rules satisfy the stronger condition that the validity of all the problems to which an ND-problem is reduced is a necessary and sufficient condition for the validity of the reduced problem. Soundness allows us to literally reduce the question of whether the given ND-problem is valid to the question of whether the problems to which it is reduced are valid. It is readily verified that all of the inference-rules for the propostional calculus correspond to sound reduction-rules.

Let us define a *simple-ND-problem* to be an ND-problem in which the conclusions and interests are all literals. Simple-ND-problems have simple solutions:

Theorem 1: If $\langle\Gamma,p\rangle$ is a simple ND-problem, $\Gamma\vDash p$ iff either (1) $p\in\Gamma$ or (2) there is a q such that $q\in\Gamma$ and $\ulcorner{\sim}q\urcorner\in\Gamma$.

In other words, a simple-ND-problem is valid iff it is solved. Completeness can thus be established by showing that the rules reduce any valid ND-problem to a finite set of simple-ND-problems.

The rules other than CONDITIONALIZATION and REDUCTIO allow us to reduce any ND-problem to a set of ND-problems in which all conclusions and interests are literals or conditionals. CONDITIONALIZATION eliminates conditionals from the interests. Thus repeated use of the rules other than REDUCTIO will reduce any ND-problem to a set of ND-problems in which the conclusions are either literals or conditionals and the interests are literals. Finally, REDUCTIO eliminates conditionals from the set of conclusions by enabling the following reduction (where r is a literal):

$$\langle\Gamma\cup\{(p\supset q)\},r\rangle$$

$$\langle\Gamma\cup\{\neg r\},{\sim}(p\supset q)\rangle$$

$$\langle\Gamma\cup\{\neg r\},p\rangle \qquad \langle\Gamma\cup\{\neg r\},\neg q\rangle$$

Thus systematic use of the rules reduces any ND-problem to a set of simple-ND-problems. Furthermore, the rules are all sound. Thus the rules allow the reduction of any valid ND-problem to a set of valid simple-ND-problems. By theorem 1, the latter problems are solved, so the reduction-tree constitutes a solution to the initial problem:

Theorem 2: The set of rules resulting from deleting MODUS PONENS, MODUS TOLLENS, and the second DISJUNCTION CONVERSION rule from the list of rules given in section one is complete.

Although MODUS PONENS, MODUS TOLLENS, and the second DISJUNCTION CONVERSION rule are not required for completeness, they play an important role in making the search for proofs more efficient.

Reduction-trees do not look exactly like the reasoning that occurs in natural deduction, but there is a simple connection between them. In performing natural deduction reasoning in the propositional calculus, at each stage of the reasoning we will have a set of conclusions $p_1,\ldots,p_n$, $q_1/X_1,\ldots,q_m/X_m$ and interests $R_1/Y_1,\ldots,R_k/Y_k$. The *embedded ND-problems* for this stage are all the ND-problems $\langle\Gamma_i,R_i\rangle$ where $\Gamma_i = Y_i\cup\{p_1,\ldots,p_n\}\cup\{Q_j \mid Y_i\subseteq X_j\}$. If the inference-rules are treated as rewrite rules, the natural deduction reasoning succeeds in establishing the desired conclusions at a particular stage of the reasoning iff every embedded ND-problem for that stage is solved.

The connection is a bit more complicated if the rules are not treated as rewrite rules. For example, consider REDUCTIO. Suppose that the conclusions drawn thus far include two conditionals,

($p \supset q$) and ($r \supset s$). Treating REDUCTIO as a rewrite rule requires restricting the rule in such a way that there is a determinate order in which it must be applied to the conditionals. We do not ordinarily treat REDUCTIO in this way. In looking for a derivation, we may try different search paths simultaneously, applying REDUCTIO to both conditionals at the same time in the hope that one of these strategies will lead to a derivation more quickly than the other. This produces two different reduction-trees, as shown in figure one. The leaf nodes of both trees will then be embedded ND-problems for the current stage of the reasoning. However, to complete the reasoning, only one of the trees has to be expanded to the point where all the leaf nodes are solved. Thus it is no longer true that a solution requires every embedded ND-problem to be solved. Instead, there will be different "sufficient sets" of ND-problems such that if all members of any sufficient set are solved then a solution has been obtained for the original problem. Interest-links and interest-schemes provide a mechanism for keeping track of different sufficient sets so that the reasoner can tell when one of them consists entirely of solved ND-problems.

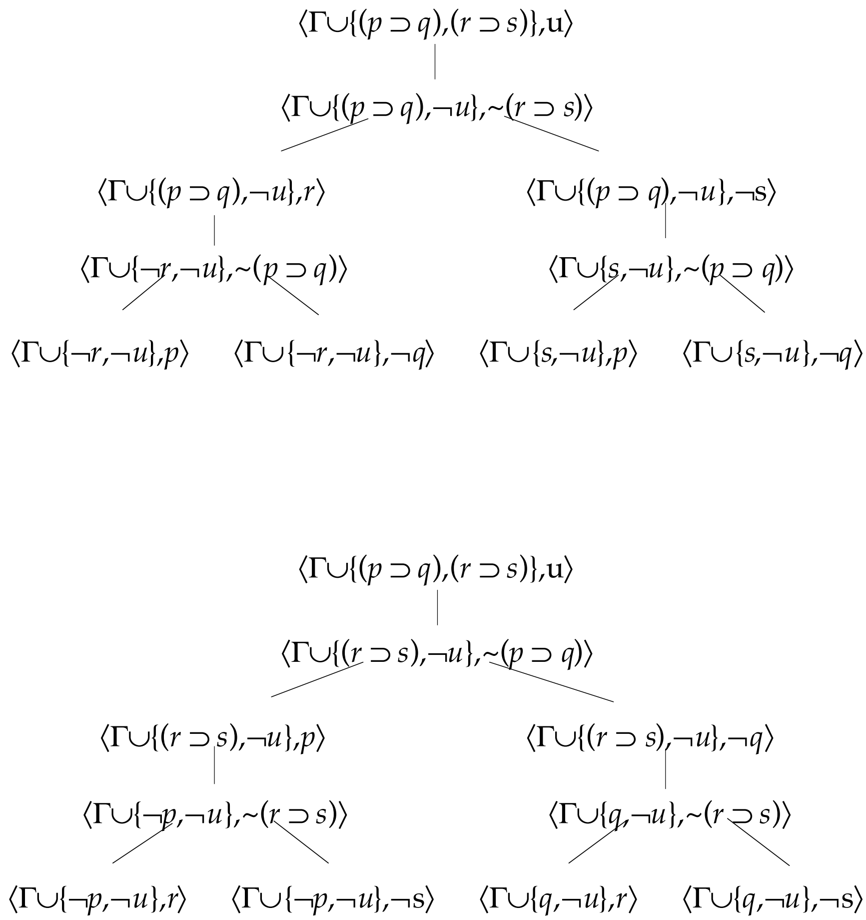

Figure 1. Two reduction-trees.

3. Skolemization and Unification in Forwards Inference

Now let us turn to natural deduction in first-order logic. The task here is to extend natural deduction for sentential logic by adding rules for handling quantifiers and variables. The natural way to do this is to use skolemization and unification. Skolemization and unification are familiar parts of the machinery of automated theorem proving. However, their use has generally been confined to systems using variants of resolution-refutation, or tableau methods, or similar methods in which the desired conclusion is negated and added to the premises, the resulting set of premises is skolemized, and then a contradiction is derived. In my [1990], I described an earlier version of OSCAR. A peculiar feature of that system was that it did not use skolemization and unification. Instead, it used "intuitive" rules of universal and existential instantiation and generalization that constructed substitution instances of quantified formulas, using all the (closed) terms that had occurred elsewhere in the argument:

<u>"Intuitive" Instantiation and Generalization Rules</u>:

universal instantiation: Where sb(x/t)p is the result of replacing free occurrences of x by t

in p, from $(\forall x)p$ infer sb$(x/t)p$ for any term t occurring elsewhere in the reasoning.

existential instantiation: From $(\exists x)p$ infer sb$(x/X_0)p$ where x_0 is a newly manufactured constant.

universal generalization: If x_0 is a constant that does not occur in either the premises or p, then from sb$(x/X_0)p$, infer $(\forall x)p$.

existential generalization: For any term t, infer $(\exists x)p$ from sb$(x/t)p$.

It has always seemed that this should be a source of considerable inefficiency, because the same reasoning will be replicated for different substitution instances. This suggests that if a version of the natural deduction system could be produced using skolemization and unification, it might be more efficient. However, it was not obvious how to use skolemization and unification in natural deduction. I will explain the difficulties, and then propose a solution to them.

Skolemizing a formula consists of putting it in prenex normal form and replacing existential quantifiers by skolem functions. The only variables remaining are then universally quantified, so the quantifiers can be dropped without ambiguity. This can also be done one step at a time, dropping initial universal quantifiers (they do not occur within the scope of existential quantifiers) and converting initial existential quantifiers into skolem-functions whose arguments are all free variables in the formula. In natural deduction, just as in resolution refutation, conclusions can be skolemized in this way. If p is a formula whose free variables are $x_1,...,x_n$ and $\sigma_1,...,\sigma_k$ are the the skolem-functions occurring in p, let SC(p) (the *skolem-closure* of p) be $(\exists\sigma_1)...(\exists\sigma_k)(\forall x_1)...(\forall x_n)p$. Then:

Theorem 3: If p is a formula and sk(p) is its skolemization, p is equivalent to SC(sk(p)).
Proof: $(\forall X_1)...(\forall X_n)(\exists y)\varphi x_1...x_n y$ is equivalent to $(\exists\sigma)(\forall x_1)...(\forall x_n)\varphi x_1...x_n\sigma(x_1...x_n)$, so the theorem is proven by induction on the order of the existential quantifiers.■

The relative order of the existential and universal quantifiers makes no difference in the skolemization.

In resolution-refutation, deductive problems are pre-processed by negating the conclusion, adding it to the list of premises, and then skolemizing the result. In natural deduction, it is more convenient to skolemize as we go along. This is done by adding the following four forwards-inference-rules (where it is assumed that distinct quantifiers contain distinct variables):

NEG-EG

$\sim(\exists x)p \Rightarrow (\forall x)\sim p$

NEG-UG

$\sim(\forall x)p \Rightarrow (\exists x)\sim p$

UI

$(\forall x)p \Rightarrow p$

EI

$(\exists x)p \Rightarrow$ sb$(x/\sigma(y_1,...,y_n))p$ where σ is a newly-constructed function symbol and $y_1,...,y_n$ are the variables having free occurrences in p.

Function symbols introduced by EI are skolem-functions. 0-adic skolem-functions are written as constants. A *skolem-term* is either a 0-adic skolem-function or an expression of the form $\sigma(t_1,...,t_n)$ where σ is an n-adic skolem-function and $t_1,...,t_n$ are terms.

In forwards inference, we can use skolemization and unification in pretty much the same way it is used in resolution refutation. I will take a unifier to be a pair of functions $\langle u_1,u_2\rangle$ instantiating free variables. $\langle u_1,u_2\rangle$ unifies p and q iff $u_1(p) = u_2(q)$. This treatment of unification allows p and q to contain free occurrences of the same variable and have it instantiated differently in the two formulas. u_1 and u_2 will be identified with a-lists of substitutions. I will take unification to be defined for sequents as well as for formulas:

> $\langle u_1,u_2\rangle$ *unifies* p/X *with* q/Y iff there is an ordering X^* of X such that $\langle u_1,u_2\rangle$ unifies $\langle p,X^*\rangle$ with $\langle q,Y\rangle$.
>
> $\langle u_1,u_2\rangle$ *unifies* p/X *into* q/Y iff there is a subset Y^* of Y such that $\langle u_1,u_2\rangle$ unifies p/X with q/Y^*.

Unification can then be used in forwards-reasoning by modifying the definition of REASON-FORWARDS-FROM. First, notice that free variables in conclusion-sequents correspond to universal quantifiers, so they can always be rewritten to ensure that different conclusions-sequents contain different free variables. This is called "writing the variables apart". We can then revise the definition of REASON-FORWARDS-FROM as follows:
as follows:

REASON-FORWARDS-FROM *conclusion*

> Given any forwards-inference-rule $\{f_1,...,f_n\} \Rightarrow f$ and any list $p_1/Y_1,...,p_n/Y_n$ of conclusion-sequents (after writing the variables apart if necessary) that includes the conclusion-sequent of *conclusion*, if $\langle u_1,u_2\rangle$ unifies the list $p_1,...,p_n$ with the list $f_1,...,f_n$, infer $u_2(f)/(u_1(Y_1) \cup ... \cup u_1(Y_n))$ from the sequents $p_1/Y_1,...,p_n/Y_n$, adding the former to the list of conclusions and placing it in the inference-queue.

This use of skolemization and unification is illustrated by the following sequence of inferences:

```
# 1
(∀x)(F x)
given
# 2
(∀x)((F x) ⊃ (G x))
given
# 3
(F x1)
from { 1 } by UI
# 4
((F x2) ⊃ (G x2))
from { 2 } by UI
# 5
(G x2)
from { 3 , 4 } by MODUS-PONENS
```

In step 5, the inference proceeds by by unifying the list ⟨(F x1) , ((F x2) ⊃ (G x2))⟩ with the list ⟨p, (P ⊃ q)⟩, taking the free variables in the former to be {x1,x2} and the free variables in the latter to be the schematic variables {p,q}. The resulting unifier is ⟨{(x1 . x2)} , {(q . (G x2)) (p . (F x2))}⟩, and the resulting conclusion is $u_2(q)$, which is (G x2).

4. Skolemization and Unification in Backwards Inference

The main problem concerning the use of skolemization and unification in natural deduction is how to use them in backwards reasoning and interest-discharge. The trick to making this work is to *reverse-skolemize* interest-formulas. The reverse skolemization of a formula simply reverses the treatment of universal and existential quantifiers. This is the same as negating the skolemization of the negation of the formula. For instance, the reverse skolemization of $(\exists x)(Fx \supset (\forall y)Gxy)$ is $(Fx \supset Gx\sigma(x))$. If p is a formula whose free variables are $x_1,...,x_n$ and $\sigma_1,...,\sigma_k$ are the the skolem-functions occurring in p, let RSC (the *reverse-skolem-closure* of p) be $(\forall\sigma_1)...(\forall\sigma_k)(\exists x_1)...(\exists x_n)p$. Then:

Theorem 4: If p is a formula and rsk(p) is its reverse-skolemization, p is equivalent to RSC(rsk(p)). Proof: By theorem 3, ~p is equivalent to SC(sk(~p)), and the latter is equivalent to ~RSC(rsk(p)), so p is equivalent to RSC(rsk(p)).■

Interests undergo reverse skolemization by adopting the following four backwards-inference-rules:

I-NEG-EG

$(\forall x)\sim p \Leftarrow \sim(\exists x)p$

I-NEG-UG

$(\exists x)\sim p \Leftarrow \sim(\forall x)p$

UG

$sb(x/\sigma(y_1,...,y_n))p \Leftarrow (\forall x)p$ where σ is a newly-constructed function symbol and $y_1,...,y_n$ are the variables having free occurrences in p.

EG

$p \Leftarrow (\exists x)p$

Skolem-functions constructed by UG will be called *i-skolem-functions* to distinguish them from skolem-functions constructed by EI. The latter will be called *c-skolem-functions*. The interest-link constructed by UG must also have an attached "discharge-condition", which will be explained shortly.

The significance of reverse-skolemization is that it is used to guide interest-discharge. 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 conclusion-formula is (F a) and the interest-formula is (∃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 (∀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 conclusion-formula is (∀x)(∃y)(∀z)(F x y z b) and the interest-formula is (∃u)(∀w)(∃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 ^@u ^w(u) ^@v) . (Typographically, the implementation constructs skolem-functions by appending '@'. It distinguishes interest-variables and interest-skolem-functions from conclusion-variables and conclusion-skolem-functions by writing the former with preceding carats: ^x or ^@x.) These unify to give us (F a y ^v b). In general, the skolemization of a conclusion-formula unifies with the reverse-skolemization of an interest-formula iff the interest-formula can be obtained from the conclusion-formula by a combination of "intuitive" universal and existential instantiation and universal and existential generalization.

Using skolemization and unification, we revise the definition of interest-discharge as follows:

A conclusion p/X discharges an interest q/Y iff p/X unifies into q/Y.

When all of the link-interests of an interest-link are "jointly discharged" (defined below), we say that the link itself is discharged, and we infer the result of applying the unifier to the link-target, adding it to the set of conclusions. For instance, the preceding reasoning was part of the following argument:

```
Given premises:
   (∀x)(F x)    justification = 1.0
   (∀x)((F x) ⊃ (G x))    justification = 1.0
Ultimate epistemic interests:
   (∀x)(G x)    interest = 1.0

 # 1
 (∀x)(F x)
 given
 # 2
  (∀x)((F x) ⊃ (G x))
  given
                          # 1
                          interest: (∀x)(G x)
                          This is of ultimate interest
                          # 2
                          interest: (G ^x0)
                          For interest 1 by UG
                          This interest is discharged by node 5
 # 3
 (F x1)
 from { 1 } by UI
```

4
((F x2) ⊃ (G x2))
from { 2 } by UI
5
(G x2)
from { 3 , 4 } by MODUS-PONENS
This discharges interest 2
6
(∀x)(G x)
from { 5 } by UG
This node is inferred by discharging interest #1

In this reasoning, conclusion 5 unifies with interest 2, discharging it and thereby discharging the interest-link between interests 1 and 2 and producing conclusion 6.

As in sentential logic, reasoning backwards produces new interests and interest-links attaching them to the target interest. However, the discharge of interest-links is made more complicated by the fact that p may unify with R and q with S while $\{p,q\}$ fails to unify with $\{R,S\}$. Some of these failures of unification may be inessential, in that they can be fixed by rewriting free variables. For instance, given the conclusions Gx/{Fx} and Hx/{Fx}, we should be able to infer (Ga & Hb)/{Fa,Fb} by discharging interest in Ga/{Fa} and Hb/{Fb}. However, {Gx,Hx} does not unify with {Ga,Hb}. Because free variables in conclusions are implicitly universally bound, we can write the variables apart so that the conclusion formulas contain different free variables. Accordingly, let us define:

> $\langle u_1,u_2\rangle$ unifies $\langle p_1/X_1 ,..., p_n/X_n\rangle$ into $\langle q_1/Y_1 ,..., q_n/Y_n\rangle$ iff, if $\langle p^*_1/X^*_1 ,..., p^*_n/X^*_n\rangle$ results from writing the free variables apart, then for each i, $\langle u_1,u_2\rangle$ unifies p^*_i/X^*_i into q_i/Y_i.

This has the desired result that ⟨Gx/{Fx} , Hx/{Fx}⟩ unifies into ⟨Ga/{Fa} , Hb/{Fb}⟩. Note, however, that individual unifiers can still fail to combine into joint unifiers. For instance, ⟨Gx/{Fxy} , Hx/{Fxy}⟩ does not unify into ⟨Ga/{Faa} , Hb/{Fab}⟩.

We can usually take a list of conclusions to discharge an interest-link if the list of conclusion-formulas unifies into the list of interest-formulas for the link-interests. However, some inference-rules require that additional conditions be satisfied. For instance, UG produces an interest in F^x from an interest in (∀x)Fx. Given a conclusion F^x/X, we cannot validly infer (∀x)Fx/X if ^x occurs in X. Thus there must be a restriction precluding that. A second restriction is required as well. Consider the following invalid reasoning:

Given premises: ----
Ultimate epistemic interests:
(∃y)(∀x)((F y) ⊃ (F x))

1
interest: (∃y)(∀x)((F y) ⊃ (F x))
This is of ultimate interest
2
interest: (∀x)((F ^@y0) ⊃ (F x))
For interest 1 by EG
This interest is discharged by node 4
3

interest: ((F ^@y0) ⊃ (F ^x1))
For interest 2 by UG
This interest is discharged by node 3

2
(F ^@y0) / { (F ^@y0) }
supposition
generated by interest 3
This discharges interest 4

4
interest: (F ^x1) / { (F ^@y0) }
For interest 3 by CONDITIONALIZATION
This interest is discharged by node 2

3
((F ^x1) ⊃ (F ^x1))
from { 2 } by CONDITIONALIZATION
This node is inferred by discharging interest #3

4
(∀x)((F ^x1) ⊃ (F x))
from { 3 } by UG
This node is inferred by discharging interest #2

5
(∃y)(∀x)((F y) ⊃ (F x))
from { 4 } by EG
This node is inferred by discharging interest #1

The inference to conclusion 4 is invalid. This can be blocked by requiring that if a conclusion discharges interest in F^x via a unifier $\langle u_1,u_2\rangle$, u_2 does not reintroduce ^x.

To accommodate such restrictions, we attach discharge-conditions to link-interests. Discharge-conditions take the arguments *conclusion unifier*. The default discharge-condition is a vacuous condition. We then say that a list of conclusions *discharges* an interest-link via $\langle u_1,u_2\rangle$ iff $\langle u_1,u_2\rangle$ unifies the list of conclusion-formulas into the list of interest-formulas for the link-interests, and for each *i*, the *i*th conclusion-formula (with rewritten variables if necessary) together with u_1 satisfies the discharge-condition (if any) for the *i*th interest. If a list of conclusions discharges an interest-link via $\langle u_1,u_2\rangle$, u_2 is applied to the target-interest and the result is inferred. Note that the target-interest itself is not inferred. Reverse-skolemized formulas are not formulas the reasoner is trying to infer. Rather, they guide the course of inference by directing the search for conclusions that unify into them. We can accordingly revise our earlier procedures as follows:

DISCHARGE-INTERESTS-FROM *conclusion*

If *conclusion* discharges *interest*, then for every interest-link whose basis contains *interest*, if *conclusion* together with some other conclusions jointly discharge the link via some unifier $\langle u_1,u_2\rangle$, infer the result of applying u_2 to the link-target, adding it to the list of conclusions and placing it in the inference-queue.

Making the discharge-condition explicit, UG should now be written as follows:

UG

$\mathrm{sb}(x/\sigma(y_1,\ldots,y_n))p \Leftarrow (\forall x)p$ where σ is a newly-constructed function symbol and $y_1,\ldots,y_n$ are the variables having free occurrences in *p*; the discharge-condition requires that if a conclusion discharges this interest via a unifier $\langle u_1,u_2\rangle$, then (1) σ does not occur in the result of applying u_1 to the conclusion-supposition, and (2) σ does not occur in u_2.

If a backwards-inference-rule does not have a discharge, then reasoning backwards in accordance with it is unchanged from such reasoning in sentential logic. However, if there is a discharge, adding the supposition $d/\{d\}$ to the list of conclusions can introduce free interest-variables and i-skolem-functions. It is not initially obvious that this is legitimate. We know how to interpret free variables and skolem-functions produced by UI and EI, but how should free interest-variables and i-skolem-functions be understood when they occur in conclusions? To sort this out intuitively, consider the simplest case of a backwards-inference-rule with a discharge—CONDITIONALIZATION. Given a reverse-skolemized interest $(p \supset q)$, we want to find a conclusion $(p^* \supset q^*)$ 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 $(p^* \supset q^*)$ unifies with $(p \supset q)$. Thus we can direct the search for $(p^* \supset q^*)$ 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 a particular 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. Accordingly, the only change required in the definition of REASON-BACKWARDS-FROM is in the clause governing discharge of the link or interest-scheme by pre-existing conclusions:

REASON-BACKWARDS-FROM *interest*

If the interest-sequent of *interest* is p/X, then given any backwards-inference-rule $\{f_1,\ldots,f_n\}\{b_1,\ldots,b_m\} \Leftarrow c$ with discharge d, if there is a pattern-match m matching c with p:

- if there are no forwards-premises,
  - if d is non-NIL:
    - add $d/\{d\}$ to the list of conclusions and put it on the inference-queue.
    - construct interests in the sequents $\mathrm{m}(f_i)/X \cup\{\mathrm{m}(d)\}$ and construct an interest-link whose resultant-interest is *interest* and whose link-interests consist of the new interests;
  - if d is NIL, construct interests in the sequents $\mathrm{m}(f_i)/X$ and construct an interest-link whose resultant-interest is *interest* and whose link-interests consist of the new interests;
  - if some list of conclusions discharges the interest-link via some unifier $\langle u_1,u_2\rangle$, infer the result of applying u_2 to *interest* from those conclusions, adding it to the list of conclusions and placing it in the inference-queue.
- if there are forwards-premises,
  - if d is non-NIL:
    - add $d/\{d\}$ to the list of conclusions and put it on the inference-queue.
    - construct an interest-scheme $\{\mathrm{m}(f_1),\ldots,\mathrm{m}(f_n)\}\{\mathrm{m}(b_1),\ldots,\mathrm{m}(b_m)\} \Leftarrow p$, with

supposition $X \cup \{m(d)\}$ and target-interest *interest*;

- if d is NIL, construct an interest-scheme $\{m(f_1),...,m(f_n)\}\{m(b_1),...,m(b_m)\} \Leftarrow p$, with supposition X and target-interest *interest*;
- if a list of conclusions $p_1/Y_1,...,p_n/Y_n$(after writing variables apart) have already been drawn that unify into $m(f_1)/X,...,m(f_n)/X$ via some unifier $\langle u_1,u_2\rangle$:
  - if there are backwards-premises, adopt interest in $u_2(m(b_1))/u_2(X),...,u_2(m(b_m))/u_2(X)$ and construct an interest-link linking those interests to *interest* and having the discharge p. If conclusions $q_1/Z_1,...,q_m/Z_m$ (after writing the variables apart) have already been drawn that unify into $u_2(m(b_1))/u_2(X),...,u_2(m(b_m))/u_2(X)$ via some unifier $\langle u_3,u_4\rangle$, infer $u_4(u_2(interest))$;
  - if there are no backwards-premises, infer $u_2(interest)$.

Similarly, DISCHARGE-INTEREST-SCHEMES-FROM is changed only in that the conclusion is computed by applying the unifier to the interest-formula:

DISCHARGE-INTEREST-SCHEMES-FROM *conclusion*

Given any interst-scheme $\{f_1,...,f_n\}\{b_1,...,b_m\} \Leftarrow c$ with discharge p and supposition X, and any list of conclusions $p_1/Y_1,...,p_n/Y_n$ of conclusion-sequents (after writing variables apart) that includes the conclusion-sequent of *conclusion*, if $p_1/Y_1,...,p_n/Y_n$ unify into $f_1/X,...,f_n/X$ via some unifier $\langle u_1,u_2\rangle$:

- if there are backwards-premises, adopt interest in $u_2(b_1)/u_2(X),...,u_2(b_m)/u_2(X)$ and construct an interest-link linking those interests to *interest* and having the discharge p. If conclusions $q_1/Z_1,...,q_m/Z_m$ (after writing the variables apart) have already been drawn that unify into $u_2(b_1)/u_2(X),...,u_2(b_m)/u_2(X)$ via some unifier $\langle u_3,u_4\rangle$, infer $u_4(u_2(interest))$;
- if there are no backwards-premises, infer $u_2(interest)$.

One change is required to the rule REDUCTIO. p must be allowed to be an existential generalization. This is because, as shown in section two, natural deduction proceeds by breaking formulas into simpler formulas, but the only way to do that with an existential generalization is by contraposing the reasoner's interests, converting existential generalizations into universal generalizations and then using UI and UG.

Here is an example that illustrates all aspects of the use of skolemization and unification in natural deduction:

Given premises:
(∀z)(∃y)(∀x)((F x y) ≡ ((F x z) & ~(F x x)))
Ultimate epistemic interests:
~(∃z)(∀x)(F x z)

1
(∀ z)(∃ y)(∀ x)((F x y) ≡ ((F x z) & ~(F x x)))
given

1
interest: ~(∃ z)(∀ x)(F x z)
This is of ultimate interest
2
interest: (∀ z)~(∀ x)(F x z)
For interest 1 by I-NEG-UG

This interest is discharged by node 22
3
interest: ~(∀ x)(F x ^x0)
For interest 2 by UG
This interest is discharged by node 21
4
interest: (∃ x)~(F x ^x0)
For interest 3 by i-NEG-UG
This interest is discharged by node 20
5
interest: ~(F ^@y ^x0)
For interest 4 by EG
This interest is discharged by node 19

3
(F ^@y ^x0)   supposition: { (F ^@y ^x0) }
supposition
generated by interest 5
This discharges interest 16
4
(∃ y)(∀ x)((F x y) ≡ ((F x x2) & ~(F x x)))
Inferred from { 1 } by UI
5
(∀ x)((F x (@y3 x2)) ≡ ((F x x2) & ~(F x x)))
Inferred from { 4 } by EI
6
((F x4 (@y3 x2)) ≡ ((F x4 x2) & ~(F x4 x4)))
Inferred from { 5 } by UI
7
((F x4 (@y3 x2)) ⊃ ((F x4 x2) & ~(F x4 x4)))
Inferred from { 6 } by BICONDITIONAL ELIMINATION
8
(((F x4 x2) & ~(F x4 x4)) ⊃ (F x4 (@y3 x2)))
Inferred from { 6 } by BICONDITIONAL ELIMINATION

8
interest: ~(((F x4 x2) & ~(F x4 x4)) ⊃ (F x4 (@y3 x2)))   supposition: { (F ^@y ^x0) }
For interest 5 by REDUCTIO
This interest is discharged by node 18
15
interest: ((F x4 x2) & ~(F x4 x4))   supposition: { (F ^@y ^x0) }
For interest 8 by CONDITIONAL NEGATION
This interest is discharged by node 17
16
interest: (F x4 x2)   supposition: { (F ^@y ^x0) }
For interest 15 by ADJUNCTION
This interest is discharged by node 3
17
interest: ~(F x4 x4)   supposition: { (F x4 ^x0) }
For interest 15 by ADJUNCTION
This interest is discharged by node 16

12
(F x4 x4)   supposition: { (F x4 x4) }

supposition
generated by interest 17

23
interest: ~(F x4 x4) supposition: { (F x4 x4) , (F x4 ^x0) }
For interest 17 by REDUCTIO
This interest is discharged by node 15

13
((F (@y3 x2) x2) & ~(F (@y3 x2) (@y3 x2))) supposition: { (F (@y3 x2) (@y3 x2)) }
Inferred from { 7 , 12 } by MODUS-PONENS
15
~(F (@y3 x2) (@y3 x2)) supposition: { (F (@y3 x2) (@y3 x2)) }
Inferred from { 13 } by SIMPLIFICATION
This discharges interest 23
16
~(F (@y3 x2) (@y3 x2))
Inferred from { 12 , 15 } by REDUCTIO
This node is inferred by discharging a link to interest #17
This discharges interest 24
17
((F (@y3 x2) ^x0) & ~(F (@y3 x2) (@y3 x2))) supposition: { (F x4 ^x0) }
Inferred from { 3 , 16 } by ADJUNCTION
This node is inferred by discharging a link to interest #15

24
interest: ~(F (@y3 x2) (@y3 ^x0)) supposition: { (F ^@y ^x0) }
For interest 8 by CONDITIONAL NEGATION
This interest is discharged by node 16

18
~(((F (@y3 ^x0) ^x0) & ~(F (@y3 ^x0) (@y3 ^x0))) ⊃ (F (@y3 ^x0) (@y3 ^x0))) supposition: { (F ^@y ^x0) }
Inferred from { 17 , 16 } by CONDITIONAL NEGATION
This node is inferred by discharging a link to interest #8
19
~(F ^@y ^x0)
Inferred from { 8 , 18 } by REDUCTIO
This node is inferred by discharging a link to interest #5
20
(∃ x)~(F x ^x0)
Inferred from { 19 } by EG
This node is inferred by discharging a link to interest #4
21
~(∀ x)(F x ^x0)
Inferred from { 20 } by i-nEG-UG
This node is inferred by discharging a link to interest #3
22
(∀ z)~(∀ x)(F x z)
Inferred from { 21 } by UG
This node is inferred by discharging a link to interest #2
23
~(∃ z)(∀ x)(F x z)
Inferred from { 22 } by I-NEG-UG
This node is inferred by discharging a link to interest #1

```
==========================================================================
 1.  (∀ z)(∃ y)(∀ x)((F x y) ≡ ((F x z) & ~(F x x)))     given
 4.  (∃ y)(∀ x)((F x y) ≡ ((F x x2) & ~(F x x)))       UI from { 1 }
 5.  (∀ x)((F x (@y3 x2)) ≡ ((F x x2) & ~(F x x)))     EI from { 4 }
 6.  ((F x4 (@y3 x2)) ≡ ((F x4 x2) & ~(F x4 x4)))     UI from { 5 }
 8.  (((F x4 x2) & ~(F x4 x4)) ⊃ (F x4 (@y3 x2)))     BICONDITIONAL ELIMINATION from { 6 }
 7.  ((F x4 (@y3 x2)) ⊃ ((F x4 x2) & ~(F x4 x4)))     BICONDITIONAL ELIMINATION from { 6 }
      |--------------------------------------------------------
      | Suppose:  { (F x4 x4) }
      |--------------------------------------------------------
      | 12.  (F x4 x4)     supposition
      |--------------------------------------------------------
      | Suppose:  { (F (@y3 x2) (@y3 x2)) }
      |--------------------------------------------------------
      | 13.  ((F (@y3 x2) x2) & ~(F (@y3 x2) (@y3 x2)))    MODUS-PONENS from { 7 , 12 }
      | 15.  ~(F (@y3 x2) (@y3 x2))     SIMPLIFICATION from { 13 }
 16.  ~(F (@y3 x2) (@y3 x2))     REDUCTIO from { 12 , 15 }
      |--------------------------------------------------------
      | Suppose:  { (F ^@y ^x0) }
      |--------------------------------------------------------
      | 3.  (F ^@y ^x0)     supposition
      |--------------------------------------------------------
      | Suppose:  { (F x4 ^x0) }
      |--------------------------------------------------------
      | 17.  ((F (@y3 x2) ^x0) & ~(F (@y3 x2) (@y3 x2)))    ADJUNCTION from { 3 , 16 }
      |--------------------------------------------------------
      | Suppose:  { (F ^@y ^x0) }
      |--------------------------------------------------------
      | 18.  ~(((F (@y3 ^x0) ^x0) & ~(F (@y3 ^x0) (@y3 ^x0))) ⊃ (F (@y3 ^x0) (@y3 ^x0)))     CONDITIONAL NEGATION
from { 17 , 16 }
 19.  ~(F ^@y ^x0)     REDUCTIO from { 8 , 18 }
 20.  (∃ x)~(F x ^x0)     EG from { 19 }
 21.  ~(∀ x)(F x ^x0)     i-NEG-UG from { 20 }
 22.  (∀ z)~(∀ x)(F x z)     UG from { 21 }
 23.  ~(∃ z)(∀ x)(F x z)     I-NEG-UG from { 22 }
==========================================================================
```

5. Soundness

In introducing the rules for using skolemization and unification in natural deduction, I have thus far relied upon loose intuitive arguments to make the procedures seem reasonable. Now I want to tighten this up. The objective is to show that if a closed formula containing no skolem-functions that is derivable from a set of closed premises containing no skolem-functions, it is a deductive consequence of those premises according the standard semantics for first-order logic. To do that, we must define a semantics for the intermediate steps of the reasoning, which typically involve free variables and skolem-functions, and prove soundness relative to that semantics.

It is initially tempting to suppose that conclusion-sequents can be regarded as short for the skolem-closures of their corresponding conditionals. Unfortunately, that will not validate standard forward inferences. For example, from the conclusions Fx@y and Gx@y we can infer (Fx@y & Gx@y). However, (∀x)(∃y)Fxy and (∀x)(∃y)Gxy do not jointly entail (∀x)(∃y)(Fxy & Gxy). To legitimate standard inferences, we must take the quantifiers to apply to all of the conclusion-sequents simultaneously. This suggests interpreting the *set* of conclusion-sequents as short for the skolem-closure of the conjunction of the corresponding conditionals of all the sequents in it. This would in fact be an adequate semantics if all of the reasoning were done in terms of forwards-inference-rules. However, it fails to validate inferences in accordance with one essential backwards-inference-rule—UG. Suppose, for instance, that the reasoner has drawn the conclusion (F^x ⊃ G^x) and from that we infer (∀x)(Fx ⊃ Gx). On the suggested semantics, we would be inferring (∀x)(Fx ⊃ Gx) from (∃σ)(Fσ ⊃ Gσ), but this inference is invalid.

The solution to this difficulty is a bit surprising. The c-skolem-functions introduced by EI and the i-skolem-functions introduced by UG work differently. The former replace existential quantifiers and the latter replace universal quantifiers. To get the semantics to work properly, if p is a formula whose free variables are $X_1,...,X_n$ and containing the i-skolem-functions $\sigma_1,...,\sigma_k$ and c-skolem-functions $\sigma_{k+1},...,\sigma_m$, let the *i-skolem-closure* of p be $(\forall\sigma_1)...(\forall\sigma_k)(\exists\sigma_{k+1})...(\exists\sigma_m)(\forall X_1)...(\forall X_n)p$. I will abbreviate this as ISC(p). The soundness theorem to be proven will be that the i-skolem-closures of all conclusions are first-order consequences of the premises. This will be proven by proving more generally that at every stage in the reasoning, the i-skolem-closure of the conjunction of the corresponding conditionals of the conclusion-sequents follows from the premises.

To prove soundness, we must make some assumptions about the inference-rules. Where p is a formula whose free variables are $X_1,...,X_n$ and containing the skolem-functions $\sigma_1,...,\sigma_k$, let $\forall p$ be $(\forall\sigma_1)...(\forall\sigma_k)(\forall X_1)...(\forall X_n)p$. I assume:

Assumption 1: If $\{f_1,...,f_n\} \Rightarrow f$ is a forwards-inference-rule, then for any substitution instance $\ulcorner\{p_1,..., p_n\} \Rightarrow p\urcorner$, $\forall[(p_1 \& ... \& p_n) \supset p]$ is valid in second-order logic.

Assumption 2: If $\{f_1,...,f_n\} \Leftarrow f$ is a backwards-inference-rule other than UG, and the rule has no discharge, then for any substitution instance $\ulcorner\{p_1,..., p_n\} \Leftarrow p\urcorner$, $\forall[(p_1 \& ... \& p_n) \supset p]$ is valid in second-order logic.

Assumption 3: If $\{f_1,...,f_n\} \Leftarrow f$ is a backwards-inference-rule with discharge d, then for any substitution instance $\ulcorner\{p_1,..., p_n\} \Leftarrow p\urcorner$, if d^* is the corresponding substitution instance of d, then $\forall[(p_1 \& ... \& p_n) \supset (d \supset p)]$ is valid in second-order logic.

It is readily verified that the abo[redacted]satisfy these assumptions. In addition, UG corresponds to inferences that s[redacted]follow[redacted]g p[redacted]nciple:

UG-principle: If σ is an i-sko[redacted]on, [redacted] is a skolem term b[redacted] is a formula, X is a s[redacted]s, and σ d[redacted]ur in either X or sb(σ*[redacted] then (∃[redacted]$X \supset p) \supset ($[redacted]$(\Pi X \supset$ s[redacted]] is valid in second-ord[redacted]gic.

Th[redacted]t of con[redacted]sion-sequ[redacted]s a[redacted]he[redacted]t of premise[redacted]d as rea[redacted] grow[redacted]rming the[redacted]e $_0$,[redacted]of sequents. T[redacted]e repre[redacted] "stages" [redacted] reaso[redacted] We can n[redacted]e two th[redacted]ms by induction on the[redacted]e:

Soun[redacted]s Theor[redacted]ery i ≥ 0, the i-skolem-closure of Π_i [redacted] consequence[redacted] Π_0.

Proof: Premises co[redacted]free v[redacted]i[redacted]les or skolem-functions, so this is v[redacted]usly tru[redacted]or $_0$. Suppose it holds for $_i$. [redacted]$_{i+1}$ [redacted]ed by adding a conclusion-sequent to $_i$ [redacted]e ways:

(1) A conclusion-sequent can be added as a supposition generated by reasoning backwards from

an interest. But suppositions have the simple fo[redacted]/{p}, so i[redacted]e theorem holds before adding a supposition, it will continue to hold.

(2) A conclusion-sequent q/X can be added as a[redacted]lt of forwa[redacted]s inference in accordance with some forwards reason-schema $\{p_1,\ldots,p_n\} \Rightarrow q$. In t[redacted]se there[redacted]ll already be conclusion-sequents $p_1/X_1,\ldots,p_n/X_n$ (with variables written apart) in [redacted]some unifier $\langle u_1,u_2\rangle$ unifies $\{p_1,\ldots,p_n\}$ with $\{p_1,\ldots,p_n\}$, $q = u_2(q)$, and $X = (u_1(X_1) \cup \ldots \cup u_1(X_n))$. It is required that the universal closure of any instance of a forwards reason-schema is valid, so $\forall\{[(\Pi X_1 \supset p_1)$ & … & $(\Pi X_n \supset p_n)] \supset (\Pi X \supset p)\}$ is valid. It follows that for any combination of quantifiers q and any formula Γ,

$$q\{[\Gamma \mathbin{\&} (\Pi X_1 \supset p_1) \mathbin{\&} \ldots \mathbin{\&} (\Pi X_n \supset p_n)] \supset (\Pi X \supset p)\}$$

is valid, and hence

$$q[\Gamma \mathbin{\&} (\Pi X_1 \supset p_1) \mathbin{\&} \ldots \mathbin{\&} (\Pi X_n \supset p_n)]$$

implies

$$q[\Gamma \mathbin{\&} (\Pi X_1 \supset p_1) \mathbin{\&} \ldots \mathbin{\&} (\Pi X_n \supset p_n) \mathbin{\&} (\Pi X \supset p)].$$

In particular,

$$(\text{ISC } [\Gamma \mathbin{\&} (\Pi X_1 \supset p_1) \mathbin{\&} \ldots \mathbin{\&} (\Pi X_n \supset p_n)])$$

implies

(ISC [Γ[redacted]X_1[redacted]&[redacted]& (Π[redacted]$\supset p_n$) & ($\Pi X \supset p$)])

so if the fo[redacted] is [redacted]ar[redacted]y the[redacted]rem[redacted]s, so is the latter.

(3) A conc[redacted]n-s[redacted]nt[redacted]ca[redacted]be a[redacted]ed as a result of discharging an interest-lin[redacted]structed in accorda[redacted]wit[redacted]he[redacted]kwa[redacted]s reason-schema $\{p_1,\ldots,p_n\}\{q_1,\ldots,q_m\} \Leftarrow$ R. For all[redacted]s except UG, the rea[redacted]ng[redacted]he[redacted]ame as in (2). For UG we appeal to the *UG-princi*[redacted]uppose p/X is in [redacted]ucte[redacted]by m[redacted]king an inference from p/X by UG. The[redacted]re is an i-skolem-function [redacted]c[redacted]structed from it and occurring in p but[redacted]in sb(σ^*/X)p, where [redacted]b($\sigma*/X$)p)}. The i-skolem-closure of Π_i [redacted] i-skolem-closure of ([redacted]X[redacted]p). As σ d[redacted]s not occur in either X or sb($\sigma*/X$)p, the u[redacted]sal closure of

(∃σ)[($\Pi X \supset p$) ⊃[redacted])([redacted]⊃ sb($\sigma*/X$)p)]

is valid. But the lat[redacted]va[redacted]nt to [[redacted]σ)($\Pi X \supset p$) ⊃ ($\forall x$)($\Pi X \supset$ sb($\sigma*/X$)p)]. T[redacted]the i-skolem-closure of Π_i [redacted]⊃ sb([redacted]X)p) is entailed by the i-skolem-closure of Π_i.[redacted]is, the i-skolem-closure of Π_{i+1} [redacted]d by the premises.■

What lies behind the preceding argument is the fact that there is an asymmetry in the way in which i-skolem-functions and c-skolem-functions are introduced into conclusion-sequents. C-skolem-functions are introduced via EI to replace existentially bound variables, so in effect, they are existentially quantified when th[redacted]uced. This is respected by the i-skolem-closure semantics. i-skolem-functions, on[redacted]ther hand, [redacted]n only be introduced into the list of conclusion-sequents in two ways. They ma[redacted] introduce[redacted]n suppositions, in which case the universal closure is valid. Alternatively, th[redacted]uced in discharging an interest-link. This can happen if a free variable in one [redacted] the inference is unified with a term in the link-interests that contains an i-[redacted]-function.[redacted]owever, free variables in the conclusion-sequents are universally bound, [redacted]e can al[redacted]ke the i-skolem-function to be universally bound.

Corollary 1: If for some i ≥ 0, *S*∈ [redacted]skolem-closure of *S* is a deductive consequence of the premises.

Corollary 2: If for some i ≥ 0, *S*∈ $_i$,[redacted]tains no free variables and no skolem-functions, then *S* is a deductive consequence of the premises.

The significance of Corollary 2 is that for formulas containing no free variables and no skolem-functions, we can dispense with the i-skolem-closure semantics. If such a formula finds its way into the conclusion-sequents, then it itself follows from the premises.

When an interest-link is discharged, the conclusion drawn is not generally the same as the resultant interest-sequent of the link. We say that the conclusion "discharges" the interest, but how are they related logically? A partial answer is provided by the following theorem:

Theorem 5: If *p* is inferred by discharging an interest-link, ∀*p* entails the reverse-skolem-closure of the resultant-sequent of the link.

Proof: ∀*p* entails the instance by "intuitive" universal and existential instantiation, and the instance entails the reverse-skolem-closure by universal and existential generalization.

6. First-Order ND-Problems

We can characterize first-order natural deduction reasoning in terms of ND-problems in much the same way as that was done for the propositional calculus, although the characterization is now more complex. To begin with, the proper treatment of variables forces us to take ND-problems to consist of sequents rather than formulas. That is, an ND-problem will be a pair $\langle \Gamma, p/X \rangle$ where Γ is a set of sequents.

Next, the standard semantics for first-order logic defines '⊨' only for closed formulas. In accordance with section five, we can extend it to open formulas by defining:

If Γ is a set of formulas and *p* is a formula, and either *p* or some member of Γ is an open formula, then $\Gamma \models p$ iff $\mathrm{ISC}(\Pi\Gamma) \models \mathrm{ISC}(p \,\&\, \Pi\Gamma)$

It is convenient to extend the definition still further to sequents by identifying sequents with their corresponding conditionals.

Let us define an *open sequent* to be a sequent *p*/*X* where either *p* is an open formula or some member of *X* is an open formula. In first-order natural deduction, interests that are open sequents are not the targets of the reasoning. Rather, the objective is to draw a conclusion that unifies into the open sequent. Accordingly, let us define:

> An ND-problem $\langle\Gamma,p/X\rangle$ is *valid* iff there is a sequent p^*/X^* such that $\Gamma \vDash p^*/X^*$ and p^*/X^* unifies into p/X.

This captures the use of reverse-skolemization in interests. The only thing that unifies with a closed formula is itself, so we have the simple theorem:

Theorem 6: If p is closed, $\langle\Gamma,p\rangle$ is valid iff $\Gamma \vDash p$.

By universal instantiation, if p^*/X^* is an open sequent and $\Gamma \vDash p^*/X^*$ then Γ implies any instance of p^*/X^*. If p^*/X^* unifies into p/X via some unifier $\langle u_1,u_2\rangle$ then there is a p^{**}/X^{**} such that $u_1(p^*) = p^{**} = u_2(p)$ and $u_1(X^*) \subseteq u_2(X)$. $p^{**}/u_1(X^*)$ is then an instance of p^*, so $\Gamma \vDash p^{**}/u_1(X^*)$. As $u_1(X^*) \subseteq u_2(X)$, it is also true that $\Gamma \vDash p^{**}/u_2(X^*)$. $p^{**}/u_2(X)$ is an instance of p/X,so we have the following theorem:

Theorem 7: $\langle\Gamma,p/X\rangle$ is valid iff there is a p^{**}/X^{**} that is an instance of p/X such that $\Gamma \vDash$ p$^{**}/X^{**}$.

We can define soundness and validity for reduction-rules just as in the propositional calculus. However, in first-order logic, the inference-rules do not all correspond to sound reduction-rules. If they did the reduction procedure would constitute a decision procedure, which is impossible by Church's Theorem.

Before addressing the soundness of the reduction rules, we must consider whether they are even valid. In the propositional calculus, the validity of the inference-rules guaranteed the validity of the reduction rules, but that relationship is no longer quite so simple in first-order logic. Numbered among the backwards-inference-rules are four "binary rules": ADJUNCTION, BICONDITIONAL INTRODUCTION, DISJUNCTION NEGATION, and CONDITIONAL NEGATION. These are rules having two backwards premises. The reduction rules produced by these rules in the propositional calculus fail to be valid in first-order logic. Consider ADJUNCTION. In the propositional calculus this produced a reduction rule reducing $\langle\Gamma,(p\&q)\rangle$ to $\langle\Gamma,p\rangle$ and $\langle\Gamma,q\rangle$. The analogous rule in first-order logic would reduce $\langle\Gamma,(p\&q)/X\rangle$ to $\langle\Gamma,p/X\rangle$ and $\langle\Gamma,q/X\rangle$. However, this is not a valid reduction. The difficulty is that we could infer a p^* the unifies with p and a q^* that unifies with q, without $(p^*\&q^*)$ unifying with $(p\&q)$. This corresponds to the observation that in first-order logic we do not automatically infer $(p^*\&q^*)$ from p^* and q^* given an interest in $(p\&q)$. We first verify that $(p^*\&q^*)$ unifies with $(p\&q)$.

The OSCAR implementation of ADJUNCTION proceeds by first finding a p^*/X^* that is implied by Γ and unifies with p/X via some unifier $\langle u_1,u_2\rangle$, and then looks for a q^*/X^{**} that that is implied by Γ and unifies with $u_2(q/X)$ (rather than with q/X) via some unifier $\langle u_3,u_4\rangle$. $u_4(u_2((p\&q)/X)))$ is then an instance of $(p\&q)/X$, and $u_4(u_2((p\&q)/X))) = (u_4(u_2(p))\ \&\ u_4(u_2(q)))/u_4(u_2(X))$. $u_4(u_2(p))/u_4(u_2(X)) = u_4(u_1(p^*))/u_4(u_1(X^*)) = u_4(u_1(p^*/X^*))$. The latter is implied by p^*/X^* by universal instantiation, and hence is implied by Γ. $u_4(u_2(q))/u_4(u_2(X)) = u_3(q^*/X^{**})$, so this is also implied by Γ. Hence $u_4(u_2((p\&q)/X)))$ is implied by Γ.

We can think of a corresponding reduction rule as reducing $\langle\Gamma,(p\&q)/X\rangle$ to the *ordered pair* $\langle\langle\Gamma,p/X\rangle, \langle\Gamma,q/X\rangle\rangle$ rather than the *set* $\{\langle\Gamma,p/X\rangle, \langle\Gamma,q/X\rangle\}$. The interpretation of reduction to the ordered pair is that a solution requires first inferring a p^*/X^* that unifies with p/X via some unifier $\langle u_1,u_2\rangle$, and then inferring a q^*/X^{**} that that unifies with $u_2(q)/u_2(X)$. In effect, this reduces $\langle\Gamma,(p\&q)/X\rangle$ to an infinite set of sets $\{\langle\Gamma,p/X\rangle, \langle\Gamma,u_2(q)/u_2(X)\rangle\}$. A solution to any one of these infinite sets is a solution to $\langle\Gamma,(p\&q)/X\rangle$. This means that the reduction of $\langle\Gamma,(p\&q)/X\rangle$ to any one of the sets $\{\langle\Gamma,p/X\rangle, \langle\Gamma,u_2(q)/u_2(X)\rangle\}$ is a valid reduction, but it is not a sound reduction because the solvability of any particular one of these sets is not a necessary condition for the solvability of $\langle\Gamma,(p\&q)/X\rangle$.

Some of the other reduction rules also fail to be sound. The least interesting failure of soundness occurs in the reduction-rule corresponding to UG. That rule reduces $\langle\Gamma,(\exists x)p/X\rangle$ to $\langle\Gamma,p/X\rangle$. This rule is certainly valid, but it is not sound. For example, $\langle\varnothing,(\exists x)(\forall y)(Fx \supset Fy)\rangle$ is valid, but $\langle\varnothing,(\forall y)(Fx \supset Fy)\rangle$ is not. For this reason, existential interests must be subject to reasoning by REDUCTIO.

The more interesting failures of soundess occur in CONDITIONALIZATION and REDUCTIO. Consider CONDITIONALIZATION. Suppose $\langle\Gamma,(p \supset q)/X)\rangle$ is valid. Then by theorem 7, there is an instance $(p^* \supset q^*)/X^*$ of $(p \supset q)/X$ such that $\Gamma \vDash (p^* \supset q^*)/X^*$. It follows that $\Gamma\cup\{p^*/p^*\} \vDash q^*/X^*\cup\{p^*\}$, and $q^*/X^*\cup\{p^*\}$ is an instance of $q/X\cup\{p\}$. However, CONDITIONALIZATION reduces $\langle\Gamma,(p \supset q)/X\rangle$ to $\langle\Gamma\cup\{p/\{p\}\},q/X\cup\{p\}\rangle$, whose validity requires that $\Gamma\cup\{p/\{p\}\} \vDash q^*/X^*\cup\{p^*\}$ (where $q^*/X^*\cup\{p^*\}$ is an instance of $q/X\cup\{p\}$). Accordingly, $\langle\Gamma\cup\{p\},q\rangle$ may not be valid. This is the point of treating suppositions with free variables as schematic suppositions. Given such a supposition p, we look for a way of instantiating it (i.e., constructing p^*) so that the corresponding instance q^* of q can be inferred. Thus all that is true of CONDITIONALIZATION is that if $\langle\Gamma,(p \supset q)\rangle$ is valid then *some instance of* $\langle\Gamma\cup\{p\},q\rangle$ is valid, not that $\langle\Gamma\cup\{p\},q\rangle$ itself is valid. REDUCTIO works similary.

Because the binary rules and CONDITIONALIZATION and REDUCTIO are not sound reduction-rules, completeness cannot be proven for the predicate calculus in the same way it was proven for the propositional calculus. To prove completeness it must be shown that the application of unification resulting from that the operation of the inference-rules always produces appropriate instances of schematic suppositions for use in CONDITIONALIZATION and REDUCTIO, and appropriate instances of conclusions of use in the binary rules. Whether this is true is at this point an unsolved problem, although I conjecture that it is true and that the inference rules presented here are complete.

7. Parallelizing Reductio

Although the rules of inference described above suffice for natural deduction reasoning in the propositional and predicate calculus, it turns out that the use of REDUCTIO exhibits some inherent inefficiencies. This is illustrated by the following problem:

```
Given premises: ----
Ultimate epistemic interests:
    (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))    interest = 1.0

                              # 1
                              interest: (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))
                              This is of ultimate interest

 # 1   ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q)))    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
 supposition
                              # 2
                              interest: ~(~p ∨ ~q)    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
                              For interest 1 by CONDITIONALIZATION
```

2   (p ∨ q)   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 1 } by SIMPLIFICATION

3   ((~p ∨ q) & (p ∨ ~q))   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 1 } by SIMPLIFICATION

4   (~p ⊃ q)   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 2 } by DISJUNCTION CONVERSION

3
interest: p   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 2 by DISJUNCTION NEGATION

5   ~p   supposition: { ~p }
supposition

4
interest: ~(~p ⊃ q)   supposition: { ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 3 by REDUCTIO using node 4
5
interest: p   supposition: { ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 3 by REDUCTIO using node 5
6
interest: ~q   supposition: { ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 3 by REDUCTIO using node 6

6   q   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~p }
Inferred from { 4 , 5 } by MODUS PONENS

7   q   supposition: { q }
supposition

7
interest: ~q   supposition: { q , ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 6 by REDUCTIO using node 6
8
interest: p   supposition: { q , ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 6 by REDUCTIO using node 5
9
interest: ~(~p ⊃ q)   supposition: { q , ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 6 by REDUCTIO using node 4

8   (~p ∨ q)   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 3 } by SIMPLIFICATION

9   (p ∨ ~q)   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 3 } by SIMPLIFICATION

10
interest: ~(p ⊃ q)   supposition: { q , ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 6 by REDUCTIO using node 10
11
interest: ~(p ⊃ q)   supposition: { ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 3 by REDUCTIO using node 10

10   (p ⊃ q)   supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 8 } by DISJUNCTION CONVERSION

12
interest: ~(~p ⊃ ~q)   supposition: { q , ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 6 by REDUCTIO using node 11

13
interest: $\sim(\sim p \supset \sim q)$ supposition: { ~p , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 3 by REDUCTIO using node 11

11 (~p ⊃ ~q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 9 } by DISJUNCTION CONVERSION

14
interest: q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 2 by DISJUNCTION NEGATION using node 13

12 ~q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~p }
Inferred from { 11 , 5 } by MODUS PONENS
Node 12 discharges interest 6

13 p supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 6 , 12 } by REDUCTIO

14 ~q supposition: { ~q }
supposition

15
interest: ~(p ⊃ q) supposition: { ~q , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 14 by REDUCTIO using node 10
16
interest: ~(~p ⊃ ~q) supposition: { ~q , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 14 by REDUCTIO using node 11
17
interest: ~(~p ⊃ q) supposition: { ~q , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 14 by REDUCTIO using node 4
18
interest: q supposition: { ~q , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 14 by REDUCTIO using node 14
19
interest: p supposition: { ~q , ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
For interest 14 by REDUCTIO using node 15
Node #13 discharges interest #19

15 ~p supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~q }
Inferred from { 10 , 14 } by MODUS TOLLENS

16 q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 15 , 13 } by REDUCTIO

17 ~(~p ∨ ~q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 13 , 16 } by DISJUNCTION NEGATION

18 (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))
Inferred from { 17 } by CONDITIONALIZATION

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

It seems that this backwards reasoning from "analogous" interests could all be done in parallel. Or more accurately, it could be done just once using a "variable-reductio-supposition", and then the resulting interests could be discharged multiple times by instantiating the variable-reductio-supposition differently for different discharging conclusions. This can be implemented by marking appropriate interests as "reductio-interests". A reductio-interest will be any interest introduced by REDUCTIO or derived by reasoning backwards from another reductio-interest. We can modify the rule of REDUCTIO so that the reductio-supposition is no longer included in the interest-supposition of a reductio-interest. Thus a single reductio-interest can encode interest in the same sequent relative to many different reductio-suppositions. Let the *non-reductio-supposition* of a conclusion be the part of the conclusion's supposition that does not consist of reductio-suppositions, and let the *reductio-ancestors* be the part that does consist of reductio-suppositions. Then reductio-interests will be discharged by conclusions whose non-reductio-suppositions are contained in the interest-suppositions. More generally, let us say that a conclusion p/X *unifies into* a reductio-interest q/Y iff there is a unifier $\langle u_1,u_2\rangle$ such that $u_1(p) = u_2(q)$ and if X^* is the non-reductio-supposition of p/X then $u_1(X^*)$ unifies into $u_2(Y)$. A conclusion discharges a reductio-interest iff it unifies into it.

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

A sufficient condition for a conclusion to discharge a reductio-interest is that the conclusion unify into the interest, but that is not a necessary condition. Consider the following reasoning:

conclusions	*interests*
R	
	(P ⊃ Q)
suppose P (for CONDITIONALIZATION)	
	Q/{P}
suppose ~Q (reductio-supposition)	
	~R (reductio-interest)
	.
	. (interest-link)
	.
	S

Suppose further reasoning produces the following conclusion:

S/{P,~Q}

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

~R / {P,~Q}

Q / {P} (by REDUCTIO)

(P ⊃ Q) (by CONDITIONALIZATION)

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

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

This can be simplified by noting that all reductio-interests have the same inherited-non-reductio-supposition. This is because the list of generating-nodes of a direct-reductio-interest is taken to be the list of all reductio-suppositions. Thus we can simply compute a single list of *inherited-non-reductio-suppositions* as we go along, augmenting it as needed when new reductio-suppositions are made. Then what is required for a conclusion to discharge a reductio-interest is that the conclusion be "appropriately-related" to the interest by a unifier $\langle u_1,u_2\rangle$ in the following sense:

> $u_1(p) = u_2(q)$ and if X^* is the set-difference of the non-reductio-supposition of p / X and the list of *inherited-non-reductio-suppositions*, then $u_1(X^*)$ unifies into $u_2(Y)$.

REDUCTIO inferences are made in response to finding conclusions that discharge direct-reductio-interests, but one qualification is required. A distinction can be made between "base-reductio-suppositions" and other reductio-suppositions. A base-reductio-supposition is one that is generated from interests that are not themselves reductio-interests. Reductio-suppositions that are not base-reductio-suppositions can be generated from each other in any order, and so they can be discharged in any order. But base-reductio-suppositions must always be discharged last, after all other reductio-suppositions have been discharged. REDUCTIO inferences are then made as follows:

DISCHARGE-REDUCTIOS *conclusion*

- If *conclusion* has reductio-ancestors, then for any direct-reductio-interest *interest* to which *conclusion* is appropriately related by some unifier $\langle u_1,u_2\rangle$, let *unifiers* be the set of unifiers unifying $u_1(X^*)$ unifies into $u_2(Y)$:
- Let Y be the node-supposition of *node*.
- For each *conclusion** that generates the direct-reductio-interest slot of *interest* (i.e., *interest* is a reductio-interest in the negation of *conclusion**) 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 *conclusion**.
  - For each reductio-ancestor R of *conclusion* or *conclusion**, if either R is not a base-reductio-supposition or it is the only reductio-ancestor of *conclusion* or *conclusion**,

and the node-formula of R is p, then draw the conclusion $\neg p\ / Y \cup Y^*\text{-}\{p\}$.

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

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

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

  # 1
  conclusion 1: P
  given
  This node encodes a deductive argument.
  Non-reductio-supposition: nil
  # 2
  conclusion 2: ~P
  given
  Non-reductio-supposition: nil

                                        # 1
                                        interest: Q
                                        This is of ultimate interest
  # 3
  conclusion 3: ~Q    supposition: { ~Q }
  reductio-supposition
  Non-reductio-supposition: nil

                                        # 2
                                        reductio interest: ~P
                                        using conclusion 1
                                        # 3
                                        reductio interest: P
                                        using conclusion 2

                                     Readopting interest in:
                                        # 1
                                        reductio interest: Q
                                        This is of ultimate interest


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

To achieve completeness, a call to DISCHARGE-FORTUITOUS-REDUCTIOS must be added to the reductio-discharge rules:

DISCHARGE-FORTUITOUS-REDUCTIOS *conclusion*

For each *conclusion** whose conclusion-formula unifies with that of *conclusion* via some unifier $\langle u_1, u_2 \rangle$:

- If *conclusion* and *conclusion** both have empty node-suppositions, then infer all ultimate interests from *conclusion* and *conclusion**.
- If *conclusion* discharges some interest generated by a supposition in the list of non-reductio-suppositions of *conclusion**, and the supposition of *conclusion* is a subset of the supposition of *conclusion**, then for each supposition *sup* in the list of non-reductio-suppositions of *conclusion** and for every generated-interest *in* of *sup*, if *conclusion* is appropriately related to *in* by to the trivial unifier $\langle \varnothing, \varnothing \rangle$, then where *p* is the result of applying u_1 to the interest-formula of *in*, *sup* is the result of applying u_1 to the supposition of *conclusion*, and *sup** is the result of applying u_2 to the supposition of *conclusion**, infer $p\,/\,sup \cup sup^*$ from *conclusion* and *conclusion**.
- If *conclusion** discharges some interest generated by a supposition in the list of non-reductio-suppositions of *conclusion*, and the supposition of *conclusion** is a subset of the supposition of *conclusion*, then for each supposition *sup* in the list of non-reductio-suppositions of *conclusion* and for every generated-interest *in* of *sup*, if *conclusion** is appropriately related to *in* by the trivial unifier $\langle \varnothing, \varnothing \rangle$, then where *p* is the result of applying u_2 to the interest-formula of *in*, *sup* is the result of applying u_1 to the supposition of *conclusion*, and *sup** is the result of applying u_2 to the supposition of *conclusion**, infer $p\,/\,sup \cup sup^*$ from *conclusion* and *conclusion**.

With these changes, a natural deduction reasoner can handle REDUCTIO reasoning properly without replicating backwards reasoning. For instance, the problem with which this section began could now be done as follows:

```
Given premises:
Ultimate epistemic interests:
     (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))    interest = 1.0

                                        # 1
                                        interest: (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))
                                        This is of ultimate interest

  # 1   ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q)))    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
  supposition
                                        # 2
                                        interest: ~(~p ∨ ~q)    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
                                        For interest 1 by CONDITIONALIZATION

  # 2   (p ∨ q)    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
  Inferred from { 1 } by SIMPLIFICATION

  # 3   ((~p ∨ q) & (p ∨ ~q))    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
  Inferred from { 1 } by SIMPLIFICATION

  # 4   (~p ⊃ q)    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
  Inferred from { 2 } by DISJUNCTION CONVERSION

                                        # 3
                                        interest: p    supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
```

For interest 2 by DISJUNCTION NEGATION

5 ~p supposition: { ~p }
supposition

4
reductio interest: ~(~p ⊃ q) supposition: {((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
using node 4

5
reductio interest: ~q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
using node 6

6 q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~p }
Inferred from { 4 , 5 } by MODUS PONENS

7 q supposition: { q }
supposition

8 (~p ∨ q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 3 } by SIMPLIFICATION

9 (p ∨ ~q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 3 } by SIMPLIFICATION

6
reductio interest: ~(p ⊃ q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
using node 10

10 (p ⊃ q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 8 } by DISJUNCTION CONVERSION

11 (~p ⊃ ~q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 9 } by DISJUNCTION CONVERSION

12 ~q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~p }
Inferred from { 11 , 5 } by MODUS PONENS

13 p supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 6 , 12 } by REDUCTIO

14 ~q supposition: { ~q }
supposition

15 ~p supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) , ~q }
Inferred from { 10 , 14 } by MODUS TOLLENS

16 q supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 15 , 13 } by REDUCTIO

17 ~(~p ∨ ~q) supposition: { ((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) }
Inferred from { 13 , 16 } by DISJUNCTION NEGATION

18 (((p ∨ q) & ((~p ∨ q) & (p ∨ ~q))) ⊃ ~(~p ∨ ~q))
Inferred from { 17 } by CONDITIONALIZATION

8. Conclusions

This paper has explored the logical foundations of automated natural deduction theorem provers and shown how to incorporate skolemization and unification into such systems. It has been shown that the key to accomplishing the latter is (1) to "reverse skolemize" interests, and (2) to use suppositions generated by CONDITIONALIZATION and REDUCTIO as schematic suppositions whose free variables are instantiated in the course of reasoning in whatever ways happen to be useful. The solution described here has been implemented in OSCAR. I am only aware of three natural deduction systems other than OSCAR—Pelletier's Thinker (Pelletier 1985), Dafa's ANDP (Dafa 1996), and Rips' PSYCOP (Rips 1994). The first two systems do not employ skolemization and unification at all, nor did the version of OSCAR described in my [1990]. PSYCOP does employ skolemization and unification to a limited degree, but as Rips observes, his use of them results in a system that is "grossly" incomplete.[8] To the best of my knowledge, OSCAR is the first natural deduction system using skolemization and unification in a way that is at least not known to be incomplete. My conjecture is that OSCAR is complete, but that remains to be proven.

References

Dafa Li

1996 "An automated natural deduction proof of the formalization of the halting problem", Association for Automated Reasoning Newsletter, No. 32, 4-8.

Pelletier, Jeffrey

1985 "THINKER". *8th International Conference on Automated Deduction.*, ed. J. H. Siekman. New York: Springer.

Pollock, John

1990 "Interest-driven suppositional Reasoning", *Journal of Automated Reasoning* **6**, 419-462.

1995 *Cognitive Carpentry*, Cambridge, Mass: Bradford/MIT Press.

1996 "Implementing defeasible reasoning". Computational Dialectics Workshop at the International Conference on Formal and Applied Practical Reasoning, Bonn, Germany, 1996. This can be downloaded from http://www.u.arizona.edu/~pollock/.

Rips, Lance

1994 *The Psychology of Reasoning*, Cambridge, Mass: Bradford/MIT Press.

Sutcliffe, G.

1996 "Results of the CADE-13 ATP system competition", *Association for Automated Reasoning Newsletter* **34**, 1-2.

Suttcliffe, G., Suttner, C.B., and Yemenis, T.

1994 "The TPTP problem library", in *Proceedings of the 12th International Conference on Automated Deduction*, ed. Alan Bundy, 252-266, Springer.

[8] Rips [1994], 215ff. It should be mentioned that Rips designed PSYCOP as a psychological model of human reasoning, and not in the attempt to build a complete first-order theorem prover.