INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106

75-25,846

BROWN, Thomas Carl, Jr., 1944-
A STRUCTURED DESIGN-METHOD FOR SPECIALIZED PROOF PROCEDURES.

California Institute of Technology, Ph.D., 1975
Mathematics

Xerox University Microfilms, Ann Arbor, Michigan 48106

© 1975

THOMAS CARL BROWN, Jr.

ALL RIGHTS RESERVED

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED.

A STRUCTURED DESIGN-METHOD FOR SPECIALIZED PROOF PROCEDURES

Thesis by

Thomas Carl Brown, Jr.

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1975

(Submitted December 4, 1974)

ACKNOWLEDGEMENTS

I wish to thank Ruth Stratton for her excellent typing work on this and earlier reports. I wish to thank Margaret Brown and my parents for their enduring love and support during this project.

I wish to thank Dr. Giorgio Ingargiola for his generosity in advising this project through to its conclusion and for his many remarks on content and style of this report. I wish to thank Dr. David Luckham for his occasional research guidance and an introduction to interactive theorem proving.

This research has been supported by NSF Grants GZ-785 and GJ28424, and by NIH Grant GM01335.

... All the time I design programs for nonexisting machines and add: "if we now had a machine comprising the primitives here assumed, then the job is done".

... In actual practice, of course, this ideal machine will turn out not to exist, so our next task--structurally similar to the original one--is to program the simulation of the "upper" machine....But this bunch of programs is written for a machine that in all probability will not exist, so our next job will be to simulate it in terms of programs for a next lower level machine, etc., until finally we have a program that can be executed by our hardware.

---E. W. Dijkstra[1]

---

[1]Notes on Structured Programming, in Structured Programming, edited by O. J. Dahl and C. A. R. Hoare (Academic Press, New York, 1972), pp. 1-81.

ABSTRACT

A proof procedure verifies relative consequence relations

$$\mathcal{B} \models_{\mathcal{E}} C \qquad (1)$$

in first-order logic with equality by generating a refutation (or proof of contradiction) for some clause-representation $\mathcal{C}$ of $\mathcal{B} \cup \{\neg C\}$, using the axioms and inferences of some sound and effective calculus for $\models_{\mathcal{E}}$. Performance of the procedure depends upon two forms of heuristic knowledge about $\models_{\mathcal{E}}$ which it may embody:

(S) Structural knowledge is formalized by a refinement (or decidable subset) of the deductions admitted by the procedure's calculus which acts as a "search-space filter": only those deductions from $\mathcal{C}$ contained in the refinement are generated.

(P) Procedural knowledge is formalized by a search strategy (or enqueuing function): it determines which of the admissible inferences will be generated next on the basis of the current deduction.

This investigation develops a general hierarchical method for the design of refinements embodying structural forms of heuristic knowledge characteristic of expert human problem solvers in an axiomatized problem domain.

Initially we design a refinement Δ for $\mathcal{E}$-resolution deductions, whose inferences have the form

$$\{B_1 \vee \underline{q}_1, \cdots, B_n \vee \underline{q}_n\} \vdash C \qquad (2)$$

where $(B_1 - q_1)\theta \vee \cdots \vee (B_n - q_n)\theta \geq C \geq (B_1\theta - q_1\theta) \vee \cdots \vee (B_n\theta - q_n\theta)$ and θ is a substitution (of terms for variables) which makes $\{q_1\theta, \cdots, q_n\theta\}$ contradictory in $\mathcal{E}$. The unit-clause set $\{q_1, \cdots, q_n\}$ is called a latent $\mathcal{E}$-contradiction.

$\mathcal{E}$-resolution is not in general effective: each inference (2) must be realized by finding a "lower level" refutation for $\mathcal{E} \cup \{q_1, \cdots, q_n\}$ and extracting θ from it. For this sub-problem we design an $\mathcal{E}'$-resolution refinement Δ' where $\mathcal{E} \supset \mathcal{E}'$. The normal composition $\Delta \cdot \Delta'$ consists of deductions in Δ wherein each inference (2) is "realized" by a refutation in Δ'; $\Delta \cdot \Delta'$ is actually an $\mathcal{E}'$-resolution refinement.

Iterating the above (with $\mathcal{E}'$ for $\mathcal{E}$), we obtain an $\mathcal{E}_0$-resolution refinement $\Delta_M = (\cdots(\Delta_n \cdot \Delta_{n-1}) \cdots \Delta_0)$ where Δ_k is an $\mathcal{E}_k$-resolution refinement and $\mathcal{E} = \mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ = unit clauses of $\mathcal{E}$. An $\mathcal{E}_0$-resolution inference is realized by refuting a latent $\mathcal{A}$-contradiction $\{p, \tilde{q}\}$ where $\mathcal{A}$ is a set of equations including $\mathcal{E}_0 \cup \{[x=x]\}$. For this sub-problem we design an ($\mathcal{E}_0$-) resolution micro-refinement Δ_μ for the set of deductions composed of factoring, binary resolution, and paramodulation inferences.

Normal refinements $\Delta_M \cdot \Delta_\mu$ combine the composite structural knowledge embodied in Δ_M with the effectiveness, efficiency, and most-general inference properties of Δ_μ.

Hyper-$\mathcal{E}$-resolution (HR($\mathcal{E}$,$\succ$,s)) exemplifies $\mathcal{E}$-resolution refinements. It relativizes to $\mathcal{E}$ a previously investigated refinement known as hyper-E-resolution with literal-ordering ($\succ$) and renaming (s).

Theorem A. Suppose that $\Delta_k = HR(\mathcal{E}_k, \succ_k, s_k)$ where each clause of $\mathcal{E}_k$ contains at most one "positive" literal under the renaming r_k ($k=0,\cdots,n$). Then Δ_M is refutation complete.

$\mathcal{E}$-normal deduction (ND($\mathcal{E}$,$\succ$)) exemplifies Δ_μ. $\succ$ is an "invariant complexity ordering" which well-orders constant terms. For each resolution inference $\{A \vee \underline{p},\ B \vee \underline{\tilde{q}}\} \vdash (A \vee B)\theta$ or paramodulation inference $\{A \vee \underline{[s=t]},\ B \vee \underline{q[r]}\} \vdash (A \vee B \vee q[t])\theta$ in a member of ND($\mathcal{E}$,$\succ$), underlined literals must have been reduced to a "least complex" normal form by a chain of $\succ$-ordered replacement operations based on equations of $\mathcal{E}$ and current derived equations; moreover, $t\theta$ cannot be "more complex" than $s\theta$. "Functional reflexivity" equations $[fx_1\cdots x_n = fx_1\cdots x_n]$, being subsumed by $[x=x]$, are excluded from $\mathcal{E}$-normal deductions by strong subsumption-deletion constraints.

Theorem B. ND($\mathcal{E}$,$\succ$) is refutation complete on unit-clause sets.

Corollary. If $\mathcal{E}$ and Δ_M are as in Theorem A and no non-unit clause of $\mathcal{E}$ contains a (positive) equation then $\Delta_M \cdot ND(\mathcal{E}_0, \succ)$ is refutation complete on clause-sets whose non-unit clauses each contain at most one equation.

Normal refinements are illustrated in the solutions of several refutation problems in Group Theory and Integer Arithmetic, where useful normal forms and complexity orderings are employed.

CONTENTS

1. THE DESIGN PROBLEM FOR REFINEMENTS 1

1.1 Introduction 2

1.1.1 Deductive Problem Solving Systems 2

1.1.2 Performance and Behavior Parameters of Proof Procedures 3

1.1.3 What Does the Human Specialist Know? 5

1.1.4 How Can Heuristic Knowledge be Formalized? 13

1.2 Predicate Logic with Equality 17

1.2.1 First-Order Vocabularies 18

1.2.2 Terms 18

1.2.3 Formulas 19

1.2.4 Occurrences in Terms and Clauses 22

1.2.5 Substitutions 23

1.2.6 Interpretations 25

1.2.7 Logical and Relative Consequence 28

1.2.8 Models and Congruence Relations 29

1.2.9 Normal Forms and Clause Representations 32

1.3 Calculi, Refinements, and Proof Procedures 34

1.3.1 Inferences 34

1.3.2 Calculi 34

1.3.3 Deductions 36

1.3.4 Refinements 37

1.3.5 Refutation Procedures 39

1.3.6 A Complete Refutation Procedure 41

1.3.7 Analysis of Resolution-Based Deductions 46

1.3.8 Liftable Calculi and Refinements 50

2. PROBLEM SOLUTION 53

2.1 $\mathcal{E}$-resolution Refinements 56

2.1.1 $\mathcal{E}$-resolution 56

2.1.2 Hyper-$\underline{\mathcal{E}}$-resolution 57

2.2 A Basic Calculus 61

2.2.1 Unification and Simplest Unifiers 61

2.2.2 Basic Inferences 63

2.2.3 Normal Embedding Transformations 65

2.2.4 Analysis of Normal Deductions 65

2.3 Resolution Micro-Refinements 68

2.3.1 Positional Order of Replacements in Derivations 68

2.3.2 Invariant Relations on Terms and Clauses 70

2.3.3 Reducibility Relations and Determinative Systems 70

2.3.4 Reduction Systems and Ordered Normal Forms 74

2.3.5 Complexity Orderings and Weighting Functions 77

2.3.6 $\underline{\mathcal{E}}$-normal Reductions 79

2.3.7 $\underline{\mathcal{E}}$-normal Inferences 80

2.3.8 $\underline{\mathcal{E}}$-normal Deductions 82

2.4 Normal Refinements and Their Proof Procedures 85

2.4.1 Normal Compositions 85

2.4.2 Normal Refinements 87

2.4.3 Normal Refutation Procedures 88

3. COMPLETENESS RESULTS 95

3.1 Completeness of $\mathcal{E}$-resolution Refinements 96

3.1.1 Horn Systems and Hyper-$\underline{\mathcal{E}}$-resolution 97

3.1.2 A Ground Completeness Proof Schema 98

3.1.3 Strong Liftability of Hyper-$\underline{\mathcal{E}}$-Resolution 100

3.2 Completeness of Resolution Micro-Refinements 103

3.2.1 Unit Completeness of ND($\mathcal{E}$,$\succ$) 103

3.2.2 Completeness Properties of Closed Reduction Systems 104

3.2.3 Convergence of an $\underline{\mathcal{E}}$-normal Derivation Procedure 108

3.3 Properties Preserved under Normal Composition 113

4. RELATED RESEARCH 125

4.1 Overviews and Textbooks 125

4.2 Formal Foundations and Background 126

4.3 Research on Equational Simplification Refinements 129

4.3.1 Studies in Combinatory Logic 129

4.3.2 Completeness of Reduction Systems Based on Complexity Orderings 130

4.3.3 Refinements Based on Composition with $\mathcal{E}$-Canonical Mappings 131

4.3.4 Demodulation and Simplification Strategies 134

4.4 Enriched Logics and Their Calculi 135

4.5 Theorem-Proving Systems 138

4.6 Deductive Problem-Solving Languages and Systems 139

5. CONCLUSIONS 142

5.1 Accomplishments 143

5.1.1 Structured Design of Specialized Refinements 143

5.1.2 Completeness Results for Hyper-$\underline{\mathcal{E}}$-resolution 144

5.1.3 Design of Resolution Micro-Refinements 144

5.2 Limitations 145

5.3 Extensions and Applications 148

5.3.1 Preliminary Empirical Investigations 148

5.3.2 Investigation of Generalized $\mathcal{E}$-resolution Refinements 150

5.3.3 Formal Improvements 151

5.3.4 Extensions to Type-Structured Logics 152

5.3.5 Type-Structured Procedural Logics 159

5.3.6 Relevancy of Present Research 163

APPENDICES

A. PROOFS OF PRINCIPAL LEMMAS 165

A.1 Proofs for $\mathcal{E}$-resolution Completeness Lemmas 165
A.2 Proofs for Resolution Micro-Refinement Lemmas 168
A.3 Proofs for Normal Composition Lemmas 174

B. UTILITY-MEASURES FOR REFINEMENTS 177

B.1 A Framework for Performance Evaluation 178
B.2 Local Measures 180
B.3 Global Measures 182

C. THE DESIGN OF COMPLEXITY ORDERINGS 185

C.1 Introduction 185
C.2 Complexity Orderings Based on Weighting Functions 188
C.3 Complexity Orderings Based on Normal Mappings 195
C.4 Canonical Reduction Systems 200
C.5 A D-complexity Ordering 203

D. A NORMAL REFINEMENT FOR INTEGER ARITHMETIC 207

D.1 Integer Arithmetic 209
D.2 A Formalization of the Irrational Prime Root Theorem 211
D.3 A Normal Refinement for Integer Arithmetic 213
D.4 A Complete Refutation 215

REFERENCES 226

1. THE DESIGN PROBLEM FOR REFINEMENTS

> We are only just beginning, it seems to me, to get a feel for these growth processes based on resolution or indeed on any other analogous logical principles. I believe that there is still much to be discovered in the way of controlling the rate and direction of growth intelligently yet automatically, without disturbing the basic completeness property. I believe that there is nothing inherently conflicting in the two leading concepts--heuristic control of the process and systematic, combinatorial control of the process, and I have tried to illustrate how these two concepts overlap and merge into each other.
>
> -- J. A. Robinson (1965)[1]

The first section of this chapter describes and motivates the problem of designing good refinements for proof procedures. Proof procedures are viewed as well-defined components of deductive problem solving systems, which provide the basis for measuring their performance.

It is shown in §1.1.3 that much of the structural knowledge available to the human specialist in an axiomatized problem domain can be formally represented by refinements. In §1.1.4, the design method outlined in the Abstract is viewed as a contribution to the structured programming of specialized proof procedures.

Subsequent sections (§1.2 and §1.3) lay the foundations for a solution to the refinement design problem in Chapter 2; these are intended more for reference purposes than for detailed reading. Terms

---

[1] Heuristic and complete processes in the mechanization of theorem proving. In Systems and Computer Science, edited by John F. Hart and Satoru Takasu (University of Toronto Press, Canada, 1967), pp. 116-124.

defined therein are cross-referenced in the text: §i means "Chapter i" and §i.j means "Section j in Chapter i".

The method of solution is summarized in the overview (§2.0) for Chapter 2. Chapter 3 summarizes the main results, relegating the more technical proofs to an appendix (§A). For illustrative purposes, it may be helpful to examine §C and §D at several points in the reading of this report.

Related research--especially that which has contributed to or motivated the research described herein--is briefly reviewed in Chapter 4.

A highly readable and more comprehensive overview of deductive problem solving in relation to Artificial Intelligence (research and applications) is available in Nilsson's recent overview [55].

1.1 Introduction

1.1.1 Deductive Problem-Solving Systems

Proof procedures. In this investigation, the term proof procedure is given a somewhat specific meaning (§1.3.5). A proof procedure is essentially a (partial) computable function Π, defined on sets of clauses, such that $\Pi(\mathcal{C})$ is a deduction (§1.3.3) from $\mathcal{C}$ based on the axioms and inference rules of some effective calculus over clauses (§1.3.2). Normally this calculus is sound relative to some set $\mathcal{E}$ of clauses, and we speak of a specialized proof procedure for $\mathcal{E}$ (or $\models_{\mathcal{E}}$, §1.2.7). The usual function of such a procedure is to derive, if possible, a refutation (or proof of contradiction) from $\mathcal{C}$ using this

calculus; in this case we speak of it as a refutation procedure.

A system (whether human, mechanical, or hybrid) which periodically invokes one or more specialized proof procedures as a part of some larger problem-solving process will be referred to as a deductive problem-solving system. Automated and "on-line" theorem provers, computer program verifiers and generators, and plan-generators for robotic systems are typical examples of deductive problem-solving systems currently being developed and used [55].

Problem domains. In theory we can select a specialized proof procedure for $\mathcal{E}$ in a given deductive problem-solving system and consider the set $\mathcal{U}_{\mathcal{E}}$ of clause-sets to which this procedure would be applied by the host system under appropriate conditions: $\mathcal{U}_{\mathcal{E}}$ is the problem domain of the proof procedure.

Application Environments. The long-term behavior of the proof-procedure's host system determines a probability measure μ on $\mathcal{U}_{\mathcal{E}}$, where $\mu(\mathcal{U})$ reflects the expectation that the host system will apply the proof procedure to some clause-set in $\mathcal{U}$. An application environment $(\mathcal{U}_{\mathcal{E}}, \mu)$ provides a realistic basis for performance analysis and comparison of alternative proof procedures (§B).

1.1.2 Performance and Behavior Parameters of Proof Procedures

Given an application environment, we can formalize various measures of (expected) performance of a proof procedure in that environment. Domain of completeness, cost, efficiency, and "expected" or "average" versions of these performance measures are defined on this basis in §B.

The major obstacle to the use of proof procedures in realistic application environments, amply documented in [55] and elsewhere, has

been the poor performance of proof procedures on at least one of several performance measures. Generally speaking, refutation completeness and reasonable (at least finite) expected relevancy (or efficiency ratios) have been difficult or impossible to achieve simultaneously.

Behavior of a proof procedure of the sort described in §1.3.6 can be analyzed in terms of two parameters which determine the computation it performs when applied to a set $\mathcal{C}$ of premises:

(i) A search space $\Delta(\mathcal{C})$ of deductions (from $\mathcal{C}$) which the procedure could generate, where Δ is a refinement of the set of deductions based on the procedure's calculus; and

(ii) A search strategy, represented by an enqueuing function which determines the order in which initial or derived clauses will be inserted in the procedure's current deduction and used in the generation of new admissible inferences.

In effect, the search strategy determines which of the deductions in $\Delta(\mathcal{C})$ the proof procedure will generate, but does not prevent the computation of a complete deduction in $\Delta(\mathcal{C})$ (§1.3.4) by forever excluding an initial or derived clause from incorporation into the deduction. (See §1.3.5, "fair schedulers".) Both (i) and (ii) can significantly affect the proof procedure's expected-performance measures.

Search-strategy design, for a given search space $\Delta(\mathcal{C})$, is fairly well understood (although not at all trivial in practice). Basically, one combines a "cost function" on deductions with a heuristic "cost of completion" predictor in order to determine which clause to select from the current clause queue in order to minimize overall cost

of completing the deduction [40]. There is no real reason why this heuristic prediction function cannot utilize "advice" attached to the clauses by some other component (e.g., the human) in the proof procedure's environment.

The design of refinements, however, appears to be a potentially more fruitful area for initial investigation. It is difficult to design and efficiently implement a good search strategy for a large unstructured search space. The exponential growth rate of search-cost as a function of simplest-refutation complexity has been a notorious obstacle to the successful use of proof procedures. Only by drastically refining the search space will we enable the computer's forte, high-speed search, to be of any real assistance in deductive problem-solving applications.

Moreover, refinements are the proper place to investigate refutation completeness, which has been difficult to analyze in specialized proof procedures with "ad hoc" refinements [29].

The scope of this investigation, consequently, has been limited to the design of refinements for specialized proof procedures. Before asking which refinements to design or how to design them, it is appropriate to ask what kinds of structural heuristic knowledge are available to an expert human problem solver in an axiomatized problem domain. Then we may consider the problem of facilitating the formalization of his knowledge by means of refinements.

1.1.3 What Does the Human Specialist Know?

The design of refutation procedures for $\models_{\mathcal{E}}$ will ultimately be done in cooperation with human specialists in the theory of $\mathcal{E}$. A

good design method should facilitate the design of refinements which incorporate the proof-structuring modes exhibited by human specialists.

The following proof-structuring modes figure prominently in many problem domains:

(i) normal form representations;

(ii) semantical abstraction;

(iii) subcase analysis; and

(iv) restricted use of axioms and theorems.

Normal form representations allow us to work with one (or a small number of) representatives of each $\mathcal{E}$-equivalence class of formulas or terms. In addition to the clause-representation of formulas, $\mathcal{E}$ may admit a useful representation of terms for a given domain $\mathcal{U}_{\mathcal{E}}$. For example, suppose $\mathcal{E}$ contains the "Commutative Ring Theory" fragment $D = \{D1,\cdots,D4\}$:

D1. $[(x+y)+z = x+(y+z)]$ (+ is associative);

D2. $[(x\cdot y)\ z = x\cdot(y\cdot z)]$ ($\cdot$ is associative);

D3. $[(x+y)\cdot z = x\cdot z + y\cdot z]$ (right distribution);

D4. $[z\cdot(x+y) = x\cdot z + y\cdot z]$ (left distribution and commutativity)

By treating each of these equations as a "reduction rule" we obtain a D-normal form $NF(D, \succ_D)$ for terms (§1.2.9,§C) wherein each term is in "right-associative, fully distributed" form. It is shown in §C that in fact $NF(D, \succ_D)$ is a D-canonical form: any two D-congruent terms "reduce" to a unique term in $NF(D, \succ_D)$.

The human specialist eliminates much redundancy and recognizes many useful equality relations (mod $\mathcal{E}$) by keeping terms and formulas in an appropriate $\mathcal{E}$-normal form while searching for a proof.

Semantical abstraction refers to the strategy of initially outlining a proof based on high-level, semantically defined inferences (or lemmas), typically of the same form ($\mathcal{B} \models_{\mathcal{E}} C$) as the original problem statement. Having completed an outline, the specialist may then verify any questionable constituent inference by means of lower-level proofs.

Subcase analysis involves postponing logical disjunctions in order to construct a "case by case" refutation of $\mathcal{E}$-inconsistent sets of unit clauses. In the context of (i) and (ii), subcase analysis yields $\mathcal{E}$-resolution inferences of the form

$$\{B_1 \vee \underline{q}_1, \cdots B_n \vee \underline{q}_n\} \vdash (B_1 \vee \cdots \vee B_n)\theta \qquad (3)$$

where $\{q_1\theta, \cdots, q_n\theta\}$ is an $\mathcal{E}$-contradictory (§1.2.6) unit-clause set. It follows that

$$\{B_1 \vee \underline{q}_1)\theta, \cdots, (B_n \vee \underline{q}_n)\theta\} \models_{\mathcal{E}} (B_1 \vee \cdots \vee B_n)\theta \qquad (4)$$

Recognizing a latent $\mathcal{E}$-contradiction $\{q_1, \cdots, q_n\}$ and "solving" for θ is often a task at which the human specialist excels; he may even have developed efficient algorithms for this task. For example, $\mathcal{E}$ may axiomatize Integer Arithmetic (§D) or Ordered Fields, and the specialist may have developed fast algorithms for solving linear systems of equations and inequalities [8,37] or proving "limit-theorems" involving continuity [7].

Restricted use of axioms and theorems is exhibited by the above specialized procedures for the unit-clause sets $\{q_1,\cdots,q_n\}$, as well as by constraints on which clauses may appear in the premises of (3) and which designated literals (q_i) shall be selected from these clauses for the subcase analysis. For example, we may require that $B_i \vee \underline{q}_i$ not be (subsumed by) a clause of $\mathcal{E}'(i=1,\cdots,n)$ where $\mathcal{E}' \subseteq \mathcal{E}$, leaving most of $\mathcal{E}$ to be represented algorithmically (by linear solvers, algebraic simplifiers, etc.) instead of axiomatically.

Example. Suppose $\mathcal{E} \supseteq D_< = D \cup \{D5\}$:

D5. $[x \nless y] \vee [y \nless z] \vee [x < z]$ (Transitivity)

Let the vocabulary V contain only $\{<,+,\cdot,0,1\}$ as extra-logical constants. Now $D_<$ has a feature common to many subsystems of interest: $D_<$ is a Horn system. It follows that each smallest latent $D_<$-contradiction has the form $\{p_1,\cdots,p_{n-1},\tilde{q}\}$ where p_i is positive and q is negative: p_i must either be an equation $[s_i = t_i]$ or an inequality $[s_i < t_i]$. There is a very natural refinement $\Delta_{D_<}$ for "basic" deductions (§2.2) which has the following features:

(a) Each refutation in $\Delta_{D_<}$ has a decomposition into basic deductions realizing $D_<$-resolution inferences (3) where no axiom of $D_<$ occurs among the premises.

(b) Each unit-clause set $\{q_1,\cdots,q_n\}$ has the above form $\{p_1,\cdots,p_{n-1},\tilde{q}\}$.

(c) In essence, D5 is used only implicitly in the transitivity rule:

$$\{A \vee \underline{[s < t]},\ B \vee [u \not\doteq v]\} \vdash A\theta \vee B\theta \vee [u \not\doteq s]\theta$$

where θ is a simplest unifier (§2.2.1) for $\{t,v\}$, and s,t,u,v are in the above D-canonical form $NF(D, \succ_D)$.

(d) Paramodulation is used only in the very restricted form $Rp(D, \succ_D)$ (§2.3.7). Generally speaking, only three kinds of replacement-inferences are allowed (stated in the unit-clause case, for simplicity):

(i) $\{[t=s],[x=x]\} \vdash [s=t]$, where $t \not\doteq_D s$.

(ii) $\{[s=t],q[r]\} \vdash q[t\theta]$, where $r = s\theta >_D t\theta$ and $[r]$ is a "left-most" occurrence of a "reducible" term r in the current set of equations (in $D \cup \{q_1,\cdots,q_n\}$).

(iii) $\{[s=t],q[r]\} \vdash q[t]$ where θ is a simplest unifier of $\{r,s\}$, $t\theta \not\doteq_D s$, and each subterm of $[s=t]$ or $q[r]$ is in $NF(D, \succ_D)$. Moreover, each equation of D is <u>strictly ordered</u> by $>_D$.

It follows by Theorem B (Corollary) in the Abstract that $\Delta_{D_<}$ contains a refutation for each $D_<$-inconsistent clause-set $\mathcal{C}$ wherein no clause contains more than one positive equation. (Thus, if $\mathcal{C}$ contains the "integer domain axiom"

$$ID.\ ([x \cdot y \neq 0] \vee [x=0] \vee [y=0])$$

and $\mathcal{C} - \{ID\}$ is $D_<$-consistent, then $\Delta_{D_<}$ might conceivably filter out <u>all</u> refutations of $\mathcal{C} \cup D_<$. This is an open question.)

The present point, however, is not the completeness of some systematically designed refinement for the problem-domain of $D_<$. The point is that $\Delta_{D_<}$ is a natural refinement, fully representative of at least one human's "expertise" at solving problems in this very simple domain. The following observation may facilitate an intuitive grasp of the kind of "efficiency" achieved by this refinement on its domain:

Proposition. Let $\mathcal{B}$ be a smallest set of constant clauses in the vocabulary of $D_<$, none of which contains an equation, such that $\Delta_{D_<}$ contains a realization $\underline{\underline{\mathcal{D}}}(C)$ for a $D_<$-resolution inference $(\mathcal{B} \vdash C)$. Then C and $\underline{\underline{\mathcal{D}}}(C)$ are uniquely determined by $\Delta_{D_<}$ and $\mathcal{B}$.

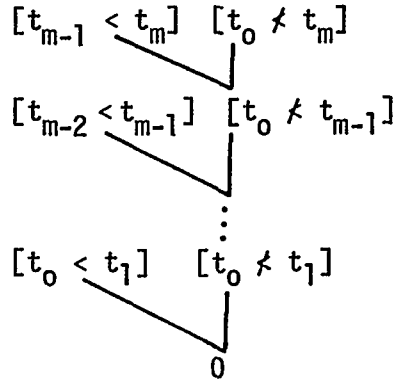
Indication of proof. $\Delta_{D_<}$ is actually a normal composition $HR(D_<,\succ_D,s) \cdot \Delta_{D5} \cdot ND(D,\succ_D)$ (§2.4) where $HR(D_<,\succ_D,s)$ defines a $D_<$-resolution refinement, Δ_{D5} restricts the use of D5 as in (c), and $ND(D,\succ_D)$ restricts the use of equations in D .

Suppose that $\Delta_{D_<}$ contains a realization $\underline{\underline{\mathcal{D}}}(C)$ for $(\mathcal{B} \vdash C)$, with premises in $\mathcal{B} \cup D_< \cup [x=x]$ and conclusion C . Then $HR(D_<,\succ_D,s)$ requires that $\mathcal{B} = \{B_1 \vee p_0, \cdots, B_{n-1} \vee \tilde{p}_{n-1}, B_n \vee \tilde{p}_n\}$ where $\{p_0,\cdots,p_{n-1},\tilde{p}_n\}$ is a $D_<$-contradiction, p_i is a maximal atom of $B_i \vee p_i$ (with respect to $\succ_D$), and $s(B_n \vee \tilde{p}_n) = \tilde{p}_n$. Thus, the conclusion $C = (B_0 \vee \cdots \vee B_n)$ is uniquely determined by the premises.

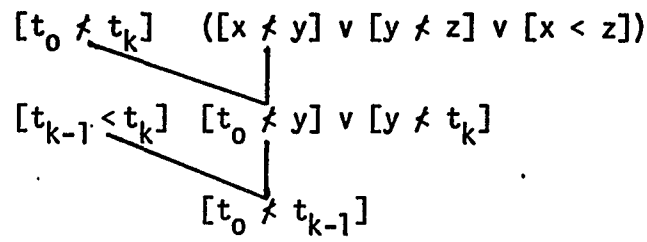
Now it suffices to show that $\{p_0,\cdots,p_{n-1},\tilde{p}_n\} \cup D_<$ has a unique refutation $\underline{\underline{\mathcal{D}}}'$ in $ND(D_<,\succ_D)$ (because $\underline{\underline{\mathcal{D}}}(C)$ is obtained by "embedding" such a refutation in $\mathcal{B} \cup D_<$).

Case 1: p_n is an equation $[u=v]$. Then $\not\vdash_{D_<} [u=v]$, because none of $\{p_0,\cdots,p_{n-1}\}$ is an equation (by assumption) and $\{p_0,\cdots,p_{n-1},\tilde{p}_n\}$ is a minimal $D_<$-contradiction (by minimality of $\mathcal{B}$). Therefore $n = 0$ and $\mathcal{B} = \{B_0\ v[u \neq v]\}$. By definition of $ND(D_<,\succ_D)$ and §C.5 it follows that $ND(D_<,\succ_D)$ contains a unique $\underline{D_<}$-normal reduction from $[u \neq v]$ to an inequality $[v' \neq v']$; this is then refuted in $\underline{\mathcal{D}}'$ by a Cut-inference $(\{[x=x],[v' \neq v']\} \vdash 0)$.

Case 2: p_n is an inequality $[u \not< v]$. It follows by minimality of $\mathcal{B}$ that $\underline{\mathcal{D}}'$ contains $n+1$ unique $\underline{D_<}$-normal reductions from members of $\{p_0,\cdots,p_{n-1},\tilde{p}_n\}$ to members of a set $\{[t_0 < t_1],\cdots,[t_{m-1} < t_m], [t_0 \not< t_m]\}$. (Thus, $[u \not< v]$ is reduced to $[t_0 \not< t_m]$ by replacement inferences based on equations of D. As in Case 1, t_i is irreducible with respect to the equations of D $(i=0,\cdots,m)$.) The completion of the refutation has the form



where $(\{[t_{k-1} < t_k], [t_0 \not< t_k]\} \vdash [t_0 \not< t_{k-1}])$ is realized by two (binary) Cut inferences,



Uniqueness of this completion follows by

(i) minimality of $\mathcal{B}$ (and hence $\{p_o,\cdots,p_{n-1},\tilde{p}_n\}$) ;

(ii) the rule-like restriction (c) imposed on the use of D5 by Δ_{D5} in $\Delta_{D_<}$; and

(iii) the requirement (actually imposed by $ND(D,\succ_D)$) that the premises of each Cut inference be irreducible with respect to D .

Thus, $\underline{\mathcal{D}}'$ and hence $\underline{\mathcal{D}}(C)$ are uniquely determined by $\Delta_{D_<}$ and $\mathcal{B}$.

Remarks.

1. There are at least $(\ell!)$ alternative refutations of the set $\{q_1,\cdots,q_n\}$ in Case 2, where ℓ is the total number of Replacement or Transitivity inferences contained in the unique refutation in $\Delta_{D_<}$. For we may think of this refutation abstractly as a "transitivity argument" (involving both $<$ and $=$) which "extends" ℓ given relations $(v_o \leq v_1,\cdots,v_{\ell-1} \leq v_\ell)$ so as to "connect" v_o to v_ℓ . There are ℓ ways of choosing a pair $\{(v_{i-1} \leq v_i), (v_i \leq v_{i+1})\}$ so as to obtain $(v_{i-1} \leq v_{i+1})$ and "eliminate v_i from consideration". The conclusion follows by induction on ℓ . A "uniform" refutation procedure using no

refinement and a pure "breadth-first" search strategy could generate each of these (ℓ!) deductions before finding the first complete refutation.

2. The "unique realization" property for $D_<$-resolution inferences on constant clauses is "essentially" preserved at the general level. For example, given a system $\{[t_0 < t_1], \cdots, [t_{n-1} < t_n]\} \cup [x \not< x]$ where $t_n = t_0$, there would be at least n distinct refutations in $\Delta_{D_<}$: one for each initial transitivity inference

$$\{[t_i < t_{i+1}], [x \not< x]\} \vdash [t_{i+1} \not< t_i].$$

1.1.4 How Can Heuristic Knowledge be Formalized?

By heuristic knowledge about an axiom system $\mathcal{E}$ (or about $\models_{\mathcal{E}}$), I refer to that knowledge or skill which aids a deductive problem-solving system (human, mechanical, or hybrid) in the solution of "relative consequence problems" $\mathcal{B} \models_{\mathcal{E}} C$.

In the context of a basic schema for proof procedures (§1.3.5) I have isolated two logically independent parameters, a refinement and a search strategy ((S) and (P) in Abstract), and have advocated the concept of an application environment (§1.1.1) as a basis for evaluation and comparison of refinements in terms of proof-procedure performance (§B).

The review of "structural" heuristic knowledge possessed by human specialists (§1.1.3) suggests a basic approach to the formalization of this knowledge in refinements. This approach should be hierarchical for several reasons:

(a) Structural heuristic knowledge about $\mathcal{E}$ is available at the non-effective, semantically defined level of $\mathcal{E}$-resolution inferences, whereas the refutation procedure operates at the (generally much lower) level of uniformly effective inferences with at most two premises.

(b) The natural sub-problem of finding refutations for $\mathcal{E}$-inconsistent unit-clause sets $\{q_1,\cdots,q_n\}$ is normally too complex to be structurally refined by the human specialist in a single step.

(c) If an effective refinement can be expressed as a composition $(\Delta_n \cdot \cdots \cdot \Delta_0) \cdot \Delta_\mu$ of increasingly lower level refinements obtained by a hierarchical design process, then it becomes feasible to obtain strong results on performance characteristics of $(\Delta_n \cdot \cdots \cdot \Delta_0) \cdot \Delta_\mu$ (domain of completeness, expected efficiency (§B), etc.) by a simpler analysis of

  (i) corresponding characteristics of the constituent refinements; and

  (ii) preservation of these characteristics under the "composition" operation $(\cdot)$.

These reasons correspond closely to several of Dijkstra's arguments in support of structured programming [18].

Structured programming, as evidenced by the quotation from Dijkstra (p. 33), is a hierarchical method for the specification of algorithms and the data structures upon which they operate [32]. The striking parallelism between the design-process for refinements

(Abstract) and the iterative process outlined by Dijkstra can be explained by viewing the refinement-design process as the data structure design component of a structured method for the specification of proof procedures. It seems appropriate to conclude this introduction by outlining this larger process of which refinement-design is a part.

The highest level in the structured programming of specialized proof procedures is exemplified by the refutation procedure Ref specified in §1.3.6. In essence, Ref has two free parameters or variables (in addition to its bound variable $\mathcal{C}$):

Δ , a refinement of the set of deductions generated by a calculus Γ ; and

Enq , representing a search strategy (actually a "fair enqueuing function") for deciding which admissible inferences to generate next in the current deduction from $\mathcal{C} \cup \mathcal{E}$ in Δ .

Merely by stipulating two axiomatic properties (finitary and fair) for Δ and Enq, we obtain a clean conceptual and functional separation of the structural knowledge (Δ) and the procedural knowledge (Enq) components of proof procedures. This separation is the basis of a simple completeness result for Ref (Proposition 6) which tells us something useful about closure properties we should consider incorporating into the design of our refinements (Proposition 5).

Now consider a normal refinement $\Delta_M \cdot \Delta_\mu$ where $\Delta_M = (\cdots(\Delta_n \cdot \Delta_{n-1}) \cdot \cdots \cdot \Delta_o)$ and Δ_k is an $\mathcal{E}_k$-resolution refinement as in the Abstract. Recall that Δ_k is designed for the subproblem of

refuting systems $\{q_1,\cdots,q_n\} \cup (\mathcal{E}_{k+1} - \mathcal{E}_k)$ where $\{q_1,\cdots,q_n\}$ is a latent $\mathcal{E}_{k+1}$-contradiction ($k=n-1,\cdots,0$). Now it seems most natural to design a search strategy E_k for the purpose of "deciding which $\mathcal{E}_k$-resolution inferences to generate next" in finding such a refutation; similarly, for Δ_μ with a search strategy E_μ (E_k and E_μ being enqueuing functions).

Using a composition operation ($\cdot$) for search strategies outlined in §2.4, we obtain a "composite" search strategy E_M for use with Δ_M and a "normal" search strategy $E_M \cdot E_\mu$ for use with $\Delta_M \cdot \Delta_\mu$.

Thus the structured programming (or design) of a refutation procedure begins with a procedure $\Pi'_n = \underline{\underline{\text{Ref}}}[\Delta_M/\Delta, E_n/\text{Enq}]$ which computes deductions based on $\mathcal{E}_n$-resolution.

Suppose we have designed $\Pi'_{k+1} = \underline{\underline{\text{Ref}}}[\Delta'_{k+1}, E'_{k+1}]$, a refutation procedure based on $\mathcal{E}_{k+1}$-resolution inferences, where $\Delta'_{k+1} = \Delta_n$ if $k+1 = n$ and $\Delta'_{k+1} = (\cdots(\Delta_n \cdot \Delta_{n-1})\cdots \Delta_{k+1})$ otherwise. We realize Π'_{k+1} by a "more effective" procedure Π'_k based on $\mathcal{E}_k$-resolution inferences as follows: $\Pi'_k = \underline{\underline{\text{Ref}}}(\Delta'_{k+1} \cdot \Delta_k, E'_{k+1} \cdot E_k)$. If $k = 0$ then we set $\Pi_M =_{df} \Pi'_0$.

Finally, we realize Π_M by an effective refutation procedure Π'_μ based on factoring, binary resolution and paramodulation in the obvious manner using Δ_μ and E_μ. In Dijkstra's words,

> We have described the program in terms of levels and each level contained "refinements" of entities that were assumed available in higher levels. These refinements were either dynamic refinements (algorithms) or static refinements

(data structures) to be understood by an appropriate machine[1].

The major data structures of a proof procedure are the deductions which it is allowed to compute (i.e., its "refinement"). The major algorithm is its enqueuing function.

The structured programming of specialized proof procedures by the methods described above has not heretofore been described or investigated. The present investigation may be regarded as a contribution to the larger task of designing useful proof procedures by structured programming methods.

1.2 Predicate Logic with Equality

A predicate logic is a certain kind of formal language with a semantics consisting of interpretations of the language into relational structures. These interpretations assign Boolean truth-values to each formula in the language.

For concreteness, in the remainder of this report we shall be concerned with a first-order predicate logic having equality as a built-in logical concept. The following paragraphs summarize aspects of this logic which are relevant to the subsequent discussion of refinements. More detailed treatments may be found in [19] and in [71].

---

[1] Notes on Structured Programming [18].

1.2.1 First-order Vocabularies

A first-order vocabulary is a set V having the following decomposition:

V_I - an infinite set of variables;

V_F^n - the set of operation constants of degree n ; V_F^0 is the set of individual constants;

V_R^n - the set of relation constants of degree n ; V_R^0 is the set of propositional constants.

V_L - $\{0,1,\sim,\vee,\wedge,\exists,\forall,=\}$ representing falsity, truth, negation, disjunction, conjunction, existential quantification, universal quantification, and equality, respectively.

Let $V_F = \cup\{V_F^n : n \in N\}$ and $V_R = \cup\{V_R^n : n \in N\}$, where N is the set of natural numbers. Let $V_E = V_F \cup V_R$. V_E is the set of extra-logical constants.

Metavariables will be used as follows: w,x,y,z for variables; $a,\cdots,f$ for operation constants; P,Q,R,S for relation constants and $=$; and h for extra-logical constants and $=$.

1.2.2 Terms

The set $\mathcal{T}_V$ of terms over V is the domain of a (totally) free algebra with generating set V_I and operations indexed by V_F . This free algebra is uniquely determined by an application function which assigns to each n+1-list $f,t_1,\cdots,t_n$ where $f \in V_F^n$ and $t_1,\cdots,t_n \in \mathcal{T}_V$, a unique object $(ft_1\cdots t_n)$ in $\mathcal{T}_V$. Thus, $\mathcal{T}_V$ is the smallest set of objects such that

(i) $V_I \cup V_F^0 \leq \mathcal{T}_V$;

(ii) If $f \in V_F^n$, $n > 0$, and $t_1,\cdots,t_n \in \mathcal{T}_V$ then $(ft_1\cdots t_n) \in \mathcal{T}_V$.

Notation. Metavariables r,s,t,u,v will vary over terms. $(ft_1\cdots t_n)$ and $f(t_1,\cdots,t_n)$ will also denote $ft_1\cdots t_n$. Moreover, if $f \in V_F^2$ then we may denote f by a commonly used infix operator (e.g., $\cdot$) and denote ft_1t_2 by, e.g., $(t_1 \cdot t_2)$.

1.2.3 Formulas

The application operation used for terms is assumed to be extended to $V_R \cup V_L$ in the manner described below, yielding a (totally) free partial algebra with domain $\mathcal{F}_V$, the formulas over V .

Atoms. The set $\mathcal{A}_V$ of atomic formulas or atoms is the smallest set of objects such that

(i) $V_R^0 \leq \mathcal{A}_V$; (Notice that the truth values 0 and 1 are not in V_R^0 .)

(ii) If $P \in V_R^n$, $n > 0$, and $t_1,\cdots,t_n \in \mathcal{T}_V$, then $(Pt_1\cdots t_n) \in \mathcal{A}_V$.

(iii) $[s=t] \in \mathcal{A}_V$ for all $s,t \in \mathcal{T}_V$.

Infix notation may be used when $P \in V_R^2$. Thus, $[s<t]$ denotes the inequality of s and t , where $<$ is a relation constant in V_R^2 .

The set $\mathcal{F}_V$ of formulas is constructed from $\mathcal{A}_V$ and V_L in the familiar manner of [71]; free and bound variables are defined,

and the sentences ($\mathscr{S}_V$) are defined to be the formulas having no free variables. The definitions

$$[A \to B] =_{df} [\sim A \vee B] \quad ;$$

$$[A \leftrightarrow B] =_{df} [(A \to B) \wedge (B \to A)] \quad ;$$

are assumed for convenience. The operation $\forall$ of universal closure sending $\mathscr{F}_V$ onto $\mathscr{S}_V$ is defined by

$$\forall(B) =_{df} \forall x_1 \cdots \forall x_n B \quad \text{where } x_1,\cdots,x_n \text{ are the free variables of } B \text{ in standard order;}$$

$$\forall(\mathcal{B}) =_{df} \{\forall(B): B \in \mathcal{B}\} .$$

Metavariables $\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{E}$ will range over sets of formulas. For the purposes of this report we need only give a precise treatment of those formulas known in the literature as literals ($\mathscr{L}_V$) and clauses ($\mathscr{C}_V$) .

The set $\mathscr{L}_V$ of literals consists of atoms and their complements. The complement of an atom p is the formula $\sim p$. Metavariables p,q, , and sometimes r,s,t,u,v will vary over $\mathscr{L}_V$. Metavariables J,K,L,M will vary over subsets of $\mathscr{L}_V$. If P is a binary relation constant (or $=$) then we define [s P t] by

$$[s\ P\ t] =_{df} \sim[s\ P\ t]$$

Complementation is defined on $\mathscr{L}_V$ by

$$\tilde{q} =_{df} \begin{cases} \sim q \ , & \text{if } q \in \mathcal{A}_V ; \\ p \ , & \text{if } q = \sim p \text{ where } p \in \mathcal{A}_V \end{cases}$$

A clause is either 0 (the empty clause), a literal (a unit clause), or a disjunction $(q_1 \vee \cdots \vee q_m)$ of m distinct literals. $(q_1 \vee \cdots \vee q_m)$ contains the literals $q_1, \cdots, q_m$ (and no others), and q contains itself. Where context permits, A, B, C, and D may be restricted to range over $\mathcal{C}_V$ without explicit mention.

Convention. Except where otherwise indicated, the order of literals in clauses is ignored, and we treat $(q_1 \vee \cdots \vee q_n)$ as denoting the logically equivalent disjunction of $q_1, \cdots, q_n$ in some standard order. The first literal of a clause A is called the designated literal of A. $A \vee p$ is the clause whose literals are those of A and p, and whose designated literal is p. This convention is essential for the representation of occurrences in clauses (§1.2.4).

$[A \vee B] =_{df}$ the clause whose literals are those of A and those of B.

Thus, $[0 \vee A] = A = [A \vee 0]$, and $[P \vee P] = P$.

$[A - B] =_{df}$ the clause whose literals are those of A less those of B.

$A \subseteq B =_{df}$ every literal of A is a literal of B.

A nonempty clause is positive if it contains only positive literals, and negative if it contains only negative literals. A non-positive clause is either empty or contains at least one negative literal.

Clauses A and B are separated provided that the variables occurring in A are disjoint from those occurring in B (i.e., A and B share no variables).

A constant term, literal, or clause is one which contains no variables.

1.2.4 Occurrences in Terms and Clauses

An occurrence in a term or atom u is represented by a pair (u,α) where α is a position in u. Positions are lists of natural numbers. Metavariables $\underline{\alpha},\underline{\beta},\underline{\gamma},\underline{\delta}$ vary over positions, and $\cdot$ denotes the concatenation operation. The empty list is denoted by $*$. The occurrence of a term t at position α in u is defined formally by

$$(t \text{ occurs in } u \text{ at } \alpha) =_{df} \begin{cases} (u = t \text{ and } \alpha = *) \text{ ; or} \\ u = (hu_1 \cdots u_n),\ \alpha = i \cdot \beta, \text{ and} \\ \quad t \text{ occurs in } u_i \text{ at } \beta . \end{cases}$$

If α is a position in u, in the sense that some term t (or atom) occurs in u at α, then we denote t by u_α. Formally, u_α is defined by

$$u_\alpha =_{df} \begin{cases} u & , \text{ if } \alpha = * \text{ ;} \\ (u_i)_\beta & , \text{ if } u = (fu_1 \cdots u_n) \text{ and } \alpha = i \cdot \beta \end{cases}$$

The set of positions in a term or atom u constitutes a (finite) tree domain D, characterized by the property that if $\alpha \cdot j \in D$ then $\alpha \in D$ and $\alpha \cdot i \in D$ $(i=0,\ldots,j-1)$.

An occurrence of u in a literal $\sim p$ at position α is represented by $(\sim p,\alpha)$ where u occurs at α in the atom (p) of $\sim p$. An occurrence of u in $A \vee q$ at position α is represented by $(A \vee q,\alpha)$ where u occurs at α in q. Thus, $(A \vee q,*)$ represents the occurrence of the atom of the designated literal q.

Notation. It is often convenient to implicitly designate an occurrence of a term or atom t by $[t]$, where $[t]$ has been used in a nearby expression denoting a term or clause. Thus, $u[t]$ is a term or literal with a designated occurrence $[u]$ at some unspecified position; similarly, $A \vee q[t]$ is a clause with a designated occurrence $[t]$ in its designated literal $q[t]$.

1.2.5 Substitutions

The instantiation of variables $x_1,\cdots,x_n$ in a term of formula u by corresponding terms $t_1,\cdots,t_n$ is the result of simultaneously replacing each free occurrence of x_i in u by t_i, and is denoted $u[t_1/x_1,\cdots,t_n/x_n]$. The mapping $\sigma: \mathcal{J}_V \cup \mathcal{F}_V \to \mathcal{J}_V \cup \mathcal{F}_V$ which carries out this instantiation operation is also denoted by $[t_1/x_1,\cdots,t_n/x_n]$, and is called a substitution.

Thus, the substitution σ such that $x_i\sigma = t_i$ $(i=1,\cdots,n)$ and $y\sigma = y$ for $y \notin \{x_1,\cdots,x_n\}$ is denoted by $[t_1/x_1,\cdots,t_n/x_n]$ or, more succinctly, by $[t_i/x_i: 1 \le i \le n]$. σ is extended to terms, clauses, and clause-sets in the obvious manner:

(i) $c\sigma = c$, for $c \in V_F^0$

(ii) $(ft_1\cdots t_n)\sigma = f(t_1)\cdots(t_n\sigma))$, for $f \in V_F^n$, $n > 0$

(iii) $(Pt_1 \cdots t_n)\sigma =_{df} P(t_1\sigma)\cdots(t_n\sigma)$, $P \in V_R^n$, $n \geq 0$

(iv) $(\sim p)\sigma =_{df} \sim(p\sigma)$

(v) $[q_1 \vee \cdots \vee q_n]\sigma =_{df} [q_1\sigma \vee \cdots \vee q_n\sigma]$

(vi) $\mathcal{B}\sigma =_{df} \{B\sigma\colon B \in \mathcal{B}\}$ . . .

Metavariables η , θ , σ , τ will vary over Σ_V , the set of substitutions on $\mathcal{T}_V$. The identity substitution is denoted by ε . $(\sigma\cdot\tau)$ denotes the composition of σ and τ , defined by

$$u(\sigma\cdot\tau) = (u\sigma)\tau \quad .$$

The substitution σ such that $x_i\sigma = t_i$ $(i=1,\cdots,n)$ and $y\sigma = y$ for $y \notin \{x_1,\cdots,x_n\}$ is denoted by $[t_1/x_1,\cdots,t_n/x_n]$, or $[t_i/x_i : 1 \leq i \leq n]$.

A substitution η is invertible provided $\eta\cdot\theta = \varepsilon$ for some θ . Each invertible substitution (other than ε) is of the form $[x_{i_1}/x_1,\cdots,x_{i_m}/x_m]$ where $(i_1,\cdots,i_m)$ is a permutation of $(1,\cdots,m)$.

Variants. Formulas B and C are variants provided $B\theta = C$ for some invertible substitution θ . Similarly, substitutions σ and τ are variants provided that $\sigma\theta = \tau$ for some invertible substitution θ .

$$\mathcal{B}^{\sim} =_{df} \{C\colon \text{ C is a variant of a member of } \mathcal{B}\} \quad .$$

Subsumption. A clause B subsumes a clause C provided that $B\sigma \subseteq C$ for some substitution σ ; if C is not a variant of B then B properly subsumes C . More generally, a formula B subsumes a formula C provided that $B\sigma = C$ for some substitution σ .

Closure of a set $\mathcal{B}$ of formulas under instantiation is denoted by * :

$$\mathcal{B}^* =_{df} \{B\sigma: B \in \mathcal{B} \text{ (and } \sigma \in \Sigma_V)\} .$$

The constant closure of $\mathcal{B}$ is denoted by $\mathcal{B}^+$:

$$\mathcal{B}^+ =_{df} \{B \in \mathcal{B}^* = B \text{ is a constant formula}\} .$$

1.2.6 Interpretations

Given a nonempty set U , we extend V to a vocabulary $V[U]$ by adjoining members of U as individual constants; $V[U]$ is just like V in §1 except for the facts that $V[U]^0_F = V^0_F \cup U$ and $V[U]^0_F$ is not necessarily disjoint from V_I . The sets $\mathcal{J}_{V[U]}$, $\mathcal{F}_{V[U]}$, $\Sigma_{V[U]}$, etc., are defined by substituting $V[U]$ for V in the definitions of $\mathcal{J}_V$, $\mathcal{F}_V$, Σ_V , etc..

An interpretation for V (with domain U) is a mapping ϕ from $V[U]_E$ to elements of U , operations on U , and relations on U such that

(a) $\phi(u) \in U$, if $u \in V_I \cup V^0_F$;

(b) $\phi(a) = a$, if $a \in U$;

(c) $\phi(P) \in \{0,1\}$, if $P \in V^0_R$;

(d) $\phi(h)$ = an operation or relation of degree n on U , if $h \in V^n_E$ $(n > 0)$.

ϕ has a standard extension to $\mathcal{J}_{V[U]} \cup \mathcal{F}_{V[U]}$ defined as follows:

(e) $\phi(ft_1 \cdots t_n) = \phi(f)(\phi t_1, \cdots, \phi t_n)$;

(f) $\phi(Pt_1\cdots t_n) = \begin{cases} 1 \ , & \text{if } (\phi t_1,\cdots,\phi t_n) \in \phi(P) \ ; \\ 0 \ , & \text{otherwise} \end{cases}$

(g) $\phi([s=t]) = \begin{cases} 1 \ , & \text{if } \phi(s) = \phi(t) \ ; \\ 0 \ , & \text{otherwise} \end{cases}$

(h) $\phi(0) = 0$; $\phi(1) = 1$; $\phi(\sim A) = 1 - \phi(A)$; $\phi(A \wedge B) = \min(\phi(A),\phi(B))$; $\phi(A \vee B) = \max(\phi(A),\phi(B))$.

(i) $\phi(\exists_y B) = \begin{cases} 1 \ , & \text{if } \phi(B[c/y]) = 1 \text{ for some } c \in U \ ; \\ 0 \ , & \text{otherwise} \end{cases}$

(j) $\phi(\forall_x B) = \begin{cases} 1 \ , & \text{if } \phi(B[c/x]) = 1 \text{ for all } c \in U \ ; \\ 0 \ , & \text{otherwise} \end{cases}$

The set U is called the domain or the universe of individuals of ϕ, and is uniquely determined by ϕ .

A formula B is satisfied by ϕ (or true under ϕ) provided that $\phi(B) = 1$; otherwise B is falsified by ϕ . B is valid under ϕ provided that $\phi(B\theta) = 1$ for all $\theta \in \Sigma_{V[U]}$; otherwise B is invalid under ϕ . Notice that B is valid under ϕ iff $\phi(\forall x_1\cdots \forall x_n B) = 1$, where $x_1,\cdots,x_n$ are the free variables of B .

A formula B is satisfiable provided B is true under some interpretation; otherwise B is unsatisfiable. B is valid provided that B is valid under every interpretation. B is consistent provided that B is valid under some interpretation; otherwise B is inconsistent.

Examples. $[x = x]$ is valid . $[x = y]$ is valid under ϕ only if the domain of ϕ has a single element. $([x \neq y] \wedge [y = z])$ is satisfiable even though it is inconsistent.

The preceding concepts extend naturally to sets of formulas. $\mathcal{B}$ is satisfied by ϕ (and hence $\mathcal{B}$ is satisfiable) provided that ϕ satisfies each formula in $\mathcal{B}$; otherwise $\mathcal{B}$ is falsified by ϕ. $\mathcal{B}$ is valid under ϕ provided that each member of $\mathcal{B}$ is valid under ϕ; otherwise $\mathcal{B}$ is invalid under ϕ. $\mathcal{B}$ is consistent provided that $\mathcal{B}$ is valid under some interpretation; otherwise $\mathcal{B}$ is inconsistent. Notice that consistency implies satisfiability, but not conversely.

$\mathcal{E}$-interpretations are those intepretations under which $\mathcal{E}$ is valid.

Conventions. The preceding concepts are relativized to the class of $\mathcal{E}$-interpretations by prefixing them with $\mathcal{E}$-. Thus, $\mathcal{B}$ is $\mathcal{E}$-satisfiable iff $\mathcal{B}$ is satisfied by an $\mathcal{E}$-interpretation, and $\mathcal{B}$ is $\mathcal{E}$-inconsistent iff $\mathcal{B}$ is invalid under every $\mathcal{E}$-interpretation.

Subsequent concepts are defined relative to (normally consistent) sets $\mathcal{E} \subseteq \mathcal{F}_V$. When $\mathcal{E}$ is the empty set, the prefix $\mathcal{E}$- is deleted.

Latent $\mathcal{E}$-contradictions. An $\mathcal{E}$-unsatisfiable set $\mathcal{C}$ is also referred to as an $\mathcal{E}$-contradiction. A set $\mathcal{B}$ is a latent $\mathcal{E}$-contradiction provided that $\mathcal{B}\sigma$ is an $\mathcal{E}$-contradiction for some $\sigma \in \Sigma_V$.

Example. The set $\mathcal{B} = \{[f(gx) \neq x], [g(fx) = x]\}$ is inconsistent because, if $\mathcal{B}$ were valid under some interpretation ϕ then $[f(g(fx)) \neq fx]$ would be true, whence $[fx = fx]$ must be true, which is impossible. Since $\mathcal{B}\sigma$ is satisfiable for all $\sigma \in \Sigma_V$, $\mathcal{B}$ is not a latent contradiction. However, the pair $\mathcal{C} = \{[f(gx) \neq x], [g(fy) = y]\}$ is a latent contradiction.

1.2.7 Logical and Relative Consequence

The relation of logical consequence holds between sets of formulas and formulas:

$(\mathcal{B} \models C) =_{df}$ C is true under every interpretation which satisfies $\mathcal{B}$;

$(\mathcal{B} \models_{\mathcal{E}} C) =_{df}$ C is true under every $\mathcal{E}$-interpretation which satisfies $\mathcal{B}$;

$(A \models_{\mathcal{E}} B) =_{df} \{A\} \models_{\mathcal{E}} B$;

$\models_{\mathcal{E}} B =_{df} \{\} \models_{\mathcal{E}} B$;

$(\mathcal{B} \models_{\mathcal{E}} \mathcal{C}) =_{df} \mathcal{B} \models_{\mathcal{E}} C$ for each $C \in \mathcal{C}$.

It is easily verified that if $\forall(\mathcal{B}) \models C$ then $\forall(\mathcal{B}) \models \forall(C)$.

The following notation for $\mathcal{E}$-equivalence will occasionally be useful:

$$(A \models\models_{\mathcal{E}} B) =_{df} A \models_{\mathcal{E}} B \text{ and } B \models_{\mathcal{E}} A .$$

Notice that if $A \models\models B$ then $\forall(A) \models\models \forall(B)$. (That the converse is not true can be seen by taking $A = (P(fx) \vee Py)$, $B = P(fx)$.)

An equality theory is a set $\mathcal{C}$ such that $\mathcal{C} = \{B: \models_{\mathcal{C}} B\}$, i.e., $\mathcal{C}$ is closed under logical consequence relative to itself. is axiomatizable provided that $\mathcal{C} = \{B: \{\} \models_{\mathcal{E}} B\}$ for some decidable set $\mathcal{E}$, in which case we refer to $\mathcal{E}$ as an axiom system (for $\mathcal{C}$) .

Notice that for every equality theory $\mathcal{C}$, either $\mathcal{C}$ is valid under some interpretation or $\mathcal{C} = \mathcal{F}_V$.

Completeness. A set $\mathcal{B} \subseteq \mathcal{I}_V$ is $\mathcal{E}$-complete for a set $\mathcal{C} \subseteq \mathcal{I}_V$ provided that either $\mathcal{B} \models_{\mathcal{E}} C$ or $\mathcal{B} \models_{\mathcal{E}} \sim C$ for each $C \in \mathcal{C}$. If $\mathcal{C} = \mathcal{I}_V$, then $\mathcal{B}$ is simply $\mathcal{E}$-complete.

Remark. The usage of complete is compatible with, but more general than, traditional usages. In the present context, it is convenient to be able to say that K is complete for $\mathcal{L}_V$ (or $\mathcal{C}_V$) provided that either $K \models p$ or $K \models \sim p$ for every $p \in \mathcal{L}_V$. Notice, incidentally, that no satisfiable set $K \subseteq \mathcal{L}_V$ can be complete for $\{\forall x(Px)\}$.

1.2.8 Models and Congruence Relations

In this report, an $\mathcal{E}$-model is an arbitrary $\mathcal{E}$-satisfiable set of literals (either in $\mathcal{L}_V$ or in $\mathcal{L}_{V[U]}$).

A congruence relation on $\mathcal{I}_V$ is an equivalence relation $\equiv$ on $\mathcal{I}_V$ such that $r \equiv s$ implies $u[r] \equiv u[s]$ $(u[w] \in \mathcal{I}_V)$. It is easily verified that the intersection of an arbitrary set of congruence relations on $\mathcal{I}_V$ is also a congruence relation on $\mathcal{I}_V$. Consequently, every set $K \subseteq \mathcal{L}_V$ determines a unique congruence relation $\equiv_K$ defined by

$$\equiv_K \ =_{df} \text{ the smallest congruence relation } \equiv \text{ on } \mathcal{I}_V \text{ such that } [s=t] \in K \text{ implies } s \equiv t .$$

$\equiv_K$ is extended to $\mathcal{L}_V$ by

$$[(Pu_1 \cdots u_n) \equiv_K (pv_1 \cdots v_n)] =_{df} u_i \equiv_K v_i \ (i=1,\cdots,n) .$$

A K-derivation (from u to v) is a list $u = u_0,\cdots,u_n = v$ such that $u_k = v_k[s_k]$ and $u_{k+1} = v_k[t_k]$ where either $[s_k = t_k] \in K$ or $[t_k = s_k] \in K$ $(k=0,\cdots,n-1)$.

Congruence compactness lemma. $u \equiv_K v$ iff there exists a K-derivation from u to v . Consequently, $u \equiv_K v$ iff $u \equiv_{K'} v$ for some finite set K' of equations in K .

Proof. It is easily verified that the relation $\sim$, defined on $\mathcal{T}_V$ by

$(u \sim v) =_{df}$ there exists a K-derivation from u to v

is a congruence relation with the property that $[s=t] \in K$ implies $s \sim t$. Thus, $\equiv_K \subseteq \sim$.

Conversely, let $\equiv$ be any congruence relation on $\mathcal{T}_V$ such that $[s=t] \in K$ implies $s \equiv t$, and suppose $u \sim v$ on the basis of the K-derivation $u = u_0,\cdots,u_n = v$ (above). Then $u_k = v_k[s_k] \equiv v_k[t_k]$ $(k=0,\cdots,n-1)$ and hence $u \equiv v$ by transitivity. Thus, $\sim = \equiv_K$ by minimality of $\equiv_K$.

Model Characterization Lemma. M is satisfiable iff

(α) $u \equiv_M v$ implies $[u \neq v] \notin M$; and

(β) $p \equiv_M q$ and $p \in M$ implies $\tilde{q} \notin M$.

Proof . Suppose ϕ satisfies M . Define $\sim_\phi$ on $\mathcal{T}_V$ by

$$(u \sim_\phi v) =_{df} \phi(u) = \phi(v) .$$

Then $\sim_\phi$ is a congruence relation on $\mathcal{J}_V$ such that $[s=t] \in M$ implies $s \sim_\phi t$, whence $\equiv_M \leq \sim_\phi$. Now (α) holds because $u \equiv_M v$ implies $\phi(u) = \phi(v)$, whence $[u \neq v] \notin M$, and (β) holds similarly.

Conversely, (α) and (β) imply that M is satisfied by ϕ_M, where

(a) $u_M =_{df}$ the first term v such that $u \equiv_M v$, according to some standard well ordering;

(b) $\phi_M(u) = u_M \quad (u \in \mathcal{J}_V)$;

(c) $\phi_M(P) = \begin{cases} 1 & , \ P \in M \ ; \\ 0 & , \ P \notin M \quad (P \in V_R^0) \ ; \end{cases}$

(d) $\phi_M(P)(u_{1_M}, \cdots, u_{n_M}) = \begin{cases} 1 & , \text{ if } (Pv_1 \cdots v_m) \in M \text{ where } u_i \equiv_M v_i \\ & \qquad\qquad (i=1, \cdots, m) \\ 0 & , \text{ otherwise } (P \in V_R^n). \end{cases}$

ϕ_M is evidently an interpretation of V with domain $\mathcal{J}_M = \{u_M : u \in \mathcal{J}_V\}$.

Model compactness lemma. Suppose K is unsatisfiable. Then some finite subset of K is unsatisfiable.

Proof. By the Model Characterization Lemma, one of the following cases must hold:

Case 1: K contains an inequation $[u \neq v]$ such that $u \equiv_K v$. Then (by the Congruence Compactness Lemma) $u \equiv_{K'} v$ for some finite subset K' of K, whence $K' \cup \{[u \neq v]\}$ is a finite unsatisfiable subset of K.

Case 2: K contains $p,\tilde{q}$ where $p \equiv_K q$. Then $p \equiv_{K'} q$ for some finite subset K' of K, whence $K' \cup \{p,\tilde{q}\}$ is a finite unsatisfiable subset of K. ∎

An equality relation (on $\mathcal{T}_V$) is a congruence relation $\equiv$ on $\mathcal{T}_V$ which is invariant under substitutions in the sense that $u \equiv v$ implies $u\theta \equiv v\theta$ $(\theta \in \Sigma_V)$. The relation $=_{\mathcal{E}}$, defined on $\mathcal{T}_V$ by

$$(u =_{\mathcal{E}} v) =_{df} (\models_{\mathcal{E}} [u = v])$$

is easily seen to be an equality relation.

Example. Let $K = \{[(x\cdot y)\cdot z = x\cdot(y\cdot z)]\}$. Then $=_K$ is the characteristic equality relation (associativity) for semigroups with operator $\cdot$, whereas $\equiv_K$ is a much smaller congruence relation on $\mathcal{T}_V$ in which, e.g., $(y\cdot x)\cdot z \not\equiv_K y\cdot(x\cdot z)$. Thus, $\equiv_K$ is not in general an equality relation.

Remark. It is easily shown that if K is a model then $=_K = \equiv_{K^*}$.

1.2.9 Normal Forms and Clause Representations

An $\mathcal{E}$-normal form for a set $\mathcal{H}$ of terms and formulas is a decidable set $\mathcal{N} \subseteq \mathcal{H}$ such that

(i) if $u \in \mathcal{H}$ then there exists $v \in \mathcal{N}$ such that $u =_{\mathcal{E}} v$;

(ii) if $A \in \mathcal{H}$ then there exists $B \in \mathcal{N}$ such that $\models_{\mathcal{E}} [A \leftrightarrow B]$

If $\mathcal{H} = \mathcal{F}_V \cup \mathcal{T}_V$ then $\mathcal{N}$ is simply an $\mathcal{E}$-normal form (for V). If u uniquely determines v in (i) and A uniquely determines B in (ii) then $\mathcal{N}$ is an $\mathcal{E}$-canonical form for $\mathcal{H}$.

An $\mathcal{E}$-normal mapping is a computable partial function ν: $\mathcal{F}_V \cup \mathcal{J}_V \rightarrow \mathcal{F}_V \cup \mathcal{J}_V$ such that

(i) $\nu(\nu(u)) = \nu(u)$ for all $u \in$ Domain(ν) ;

(ii) if $A \in$ Domain(ν) then $A \underset{\mathcal{E}}{\vdash\!\dashv} \nu(A)$;

(iii) if $u \in$ Domain(ν) then $u =_{\mathcal{E}} \nu(u)$;

(iv) Range(ν) is decidable, and is therefore an $\mathcal{E}$-normal form for Domain(ν) .

If $\nu(A) \neq \nu(B)$ implies $\nu(A) \underset{\mathcal{E}}{\vdash\!\dashv} \nu(B)$ and $\nu(u) \neq \nu(v)$ implies $u \neq_{\mathcal{E}} v$, then ν is an $\mathcal{E}$-canonical mapping.

$\mathcal{E}$-normal forms and mappings are relevant to the design of refinements for a problem-domain $\mathcal{U}_{\mathcal{E}}$ because it is desirable that only a small number of representatives of an $\mathcal{E}$-equivalence class of formulas should be derivable from given premises within the refinement.

One application of these concepts is that every finite set $\mathcal{C}$ of formulas can be represented by a finite set $\overline{\mathcal{C}}$ of clauses such that $\mathcal{C}^* \underset{\mathcal{E}(\mathcal{C})}{\vdash\!\dashv} \overline{\mathcal{C}}^*$, where $\mathcal{E}(\mathcal{C})$ is a set of Skolem axioms of the form

$$\exists y B \rightarrow B[f_{\exists y B}(x_1 \cdots x_n)/y] \qquad (2)$$

where B is a formula containing only those extra-logical constants in $\mathcal{C} \cup \mathcal{E}(\mathcal{C})$ and having free variables $x_1, \cdots, x_n$, and $f_{\exists}$ is a corresponding operation constant of degree n which does not occur in $\mathcal{C}$.[1] It is easily shown that $\mathcal{C}$ is consistent iff

[1] See [71] for details of a similar treatment of quantifier elimination.

$\mathcal{C} \cup \mathcal{E}(\mathcal{C})$ is consistent. It follows from $(\mathcal{C}^* \models_{\mathcal{E}(\mathcal{C})} \overline{\mathcal{C}}^*)$ that $\mathcal{C}$ is consistent iff $\overline{\mathcal{C}}$ is consistent. The clause-representation mapping from $\mathcal{C}$ to $\overline{\mathcal{C}}$ is a simple $\mathcal{E}(\mathcal{C})$-normal mapping; it is clearly described in [54] . Since the calculi and refinements investigated in this report operate exclusively on clauses, it is unnecessary to define the mapping here. Clause representations for several familiar axiom systems are given in the appendices.

1.3 Calculi, Refinements, and Proof Procedures

1.3.1 Inferences

An inference consists of a finite set of formulas called premises and another formula, the conclusion. We may think of an inference as an ordered pair $(\mathcal{B},C)$ or as an asserted relation $(\mathcal{B} \vdash C)$. $\mathcal{B} \cup \{C\}$ is normally a set of clauses.

An inference $(\mathcal{B} \vdash C)$ is $\mathcal{E}$-sound provided that $\forall(\mathcal{B}) \models_{\mathcal{E}} C$. An allegedly $\mathcal{E}$-sound inference $(\mathcal{B} \vdash C)$ may be represented by the relation $(\forall(\mathcal{B}) \models_{\mathcal{E}} C)$.

Example. $(\{[x=y],Pa,\sim Pb\} \vdash 0)$ is a sound inference because $\{\forall x \forall y[x=y],Pa,\sim Pb\} \models 0$.

1.3.2 Calculi

A calculus is a set Γ of formulas and inferences. Normally the formulas are clauses and the inferences consists of clauses, and we say that Γ is a calculus over clauses.

A Γ-axiom is a variant of a clause in Γ :

$$Ax(\Gamma) =_{df} \{A: A \in \Gamma\}^{\sim} .$$

The result of extending Γ to include members of $\mathcal{A}$ as axioms is denoted $\Gamma[\mathcal{A}]$:

$$\Gamma[\mathcal{A}] =_{df} \Gamma \cup \mathcal{A}$$

A <u>Γ-inference</u> is a "variant" of an inference in Γ :

$(\mathcal{B} \vdash_{\Gamma} C) =_{df} (\mathcal{B}\eta, C\eta) \in \Gamma$ for some invertible η in Σ_V .

A <u>Γ-theorem</u> is either a Γ-axiom or the conclusion of a Γ-inference whose premises are all Γ-theorems:

$Th(\Gamma) =_{df} \bigcap\{T: Ax(\Gamma) \subseteq T$ and, if $\mathcal{B} \subseteq T$ and $\mathcal{B} \vdash_{\Gamma} C$ then $C \in T\}$. $(\mathcal{B} \vdash_{\Gamma}^{*} C) =_{df} C \in Th(\Gamma[\mathcal{B}])$.

<u>Soundness</u>, <u>completeness</u>, and <u>effectiveness</u> are three basic characteristics of calculi. The following definitions are appropriate for clause-based refutation-oriented systems where universal quantification of free variables is implicit.

Γ is <u>$\mathcal{E}$-sound</u>, or <u>sound for</u> $\models_{\mathcal{E}}$, provided that (i) and (ii) hold:

(i) If $B \in Ax(\Gamma)$ then $\models_{\mathcal{E}} \forall(B)$

(ii) If $\mathcal{B} \vdash_{\Gamma} C$ then $\forall(\mathcal{B}) \models_{\mathcal{E}} C$.

Γ is <u>$\mathcal{E}$-complete</u>, or <u>refutation complete for</u> $\models_{\mathcal{E}}$, provided that (iii) holds:

(iii) If $\forall(\mathcal{C}) \models_{\mathcal{E}} 0$ then $\mathcal{C} \vdash_{\Gamma}^{*} 0$

Γ is <u>$\mathcal{E}$-adequate</u>, or <u>adequate for</u> $\models_{\mathcal{E}}$, provided that Γ is $\mathcal{E}$-sound and $\mathcal{E}$-complete.

Γ is <u>effective</u> provided that the axioms and inferences of Γ constitute a decidable set.

1.3.3 Deductions

A deduction is a partially ordered clause-set $\underline{\underline{\mathfrak{D}}} = (\mathfrak{D}, \prec)$; $\mathfrak{D}$ is normally finite.

$\underline{\underline{\mathfrak{D}}}$ is a refutation provided that $0 \in \mathfrak{D}$.

The inferences in $\underline{\underline{\mathfrak{D}}}$ are the pairs $(\mathcal{B}, C)$ where $\mathcal{B}$ is the set of immediate predecessors of C in $\prec$:

$(\mathcal{B} \underset{\underline{\underline{\mathfrak{D}}}}{\vdash} C) =_{df} (\mathcal{B}, C)$ is an inference in $\underline{\underline{\mathfrak{D}}}$.

The base of $\underline{\underline{\mathfrak{D}}}$ is the set of minimal clauses or premises of $\underline{\underline{\mathfrak{D}}}$:

$$\text{Base}(\underline{\underline{\mathfrak{D}}}) =_{df} \{B \in \mathfrak{D} : A \preceq B \text{ implies } A = B \ (A \in \mathfrak{D})\}$$

The conclusions of $\underline{\underline{\mathfrak{D}}}$ are the maximal clauses of $\underline{\underline{\mathfrak{D}}}$.

$\underline{\underline{\mathfrak{D}}}$ is a Γ-deduction (from $\mathcal{B}$) provided that

(i) $\text{Base}(\underline{\underline{\mathfrak{D}}}) \subseteq \mathcal{B} \cup \text{Ax}(\Gamma)$; and

(ii) If $\mathcal{B}' \underset{\underline{\underline{\mathfrak{D}}}}{\vdash} C'$ then $\mathcal{B}' \vdash_{\Gamma} C'$ --i.e., each inference in $\underline{\underline{\mathfrak{D}}}$ is a Γ-inference.

A Γ-refutation of $\mathcal{B}$ is a Γ-deduction from $\mathcal{B}$ which contains 0 .

A Γ-realization of an inference $(\mathcal{B} \vdash C)$ is a Γ-deduction $\underline{\underline{\mathfrak{D}}}$ from $\mathcal{B}$ such that C is a conclusion of $\underline{\underline{\mathfrak{D}}}$.

The ancestors of C in $\underline{\underline{\mathfrak{D}}}$ are the set $\mathfrak{D}(C)$ defined by

$$\mathfrak{D}(C) =_{df} \{A \in \mathfrak{D} : A \preceq C\}.$$

The deduction $\underline{\underline{\mathfrak{D}}}(C) = (\mathfrak{D}(C), \prec)$ is an initial subdeduction of $\underline{\underline{\mathfrak{D}}}$. If $\underline{\underline{\mathfrak{D}}} = \underline{\underline{\mathfrak{D}}}(C)$ then $\underline{\underline{\mathfrak{D}}}$ may be referred to as a proof tree; observe that $\text{Th}(\Gamma) = \{C$: there exists a Γ-deduction $\underline{\underline{\mathfrak{D}}}(C)$ from $\text{Ax}(\Gamma)\}$.

More generally, a subdeduction of $\underline{\mathfrak{D}} = (\mathfrak{D}, \prec)$ is a deduction $\underline{\mathfrak{D}}' = (\mathfrak{D}', \prec')$ satisfying (i)-(iii):

(i) $\mathfrak{D}' \subseteq \mathfrak{D}$.

(ii) If $\mathcal{B} \underset{\underline{\mathfrak{D}}'}{\vdash} C$ then $\mathcal{B} \underset{\underline{\mathfrak{D}}}{\vdash} C$.

(iii) If $\mathcal{B} \underset{\underline{\mathfrak{D}}}{\vdash} C$ and $A \prec C$ where $A, C \in \mathfrak{D}'$ then $\mathcal{B} \underset{\underline{\mathfrak{D}}'}{\vdash} C$.

A decomposition of $\underline{\mathfrak{D}}$ is a collection $\{\underline{\mathfrak{D}}_i(C_i) = i \leq n\}$ of subdeductions of $\underline{\mathfrak{D}}$ satisfying (i)-(iii):

(i) $\mathfrak{D} = \bigcup\{\mathfrak{D}_i(C_i): i \leq n\}$

(ii) $\text{Base}(\underline{\mathfrak{D}}_j(C_j)) \subseteq \text{Base}(\underline{\mathfrak{D}}) \cup \{C_i: i < j\}$.

(iii) If $i < j$ then $\mathfrak{D}_i(C_i) \cap \mathfrak{D}_j(C_j) \subseteq \{C_i\}$.

Thus, a decomposition of $\underline{\mathfrak{D}}$ breaks $\underline{\mathfrak{D}}$ up into a set of essentially disjoint subtrees.

1.3.4 Refinements

Let Γ be a calculus (over clauses). A (Γ-)refinement is a set Δ of (finite Γ-deductions such that

(i) Δ is decidable in Γ[1] ; and

(ii) If $\underline{\mathfrak{D}} \in \Delta$ and $\underline{\mathfrak{D}}'$ is a subdeduction of $\underline{\mathfrak{D}}$ such that $\text{Base}(\underline{\mathfrak{D}}') \subseteq \text{Base}(\underline{\mathfrak{D}})$ then $\underline{\mathfrak{D}}' \in \Delta$.

Δ may be complete relative to $\models_{\mathcal{E}}$, a calculus, or another refinement, each in an intuitively natural sense. Let Δ be a Γ-refinement and let Δ' be a Γ'-refinement.

---

[1] Decidable in (or relative to) a set means (effectively) decidable on the basis of the characteristic function of the set. Γ (e.g., $\mathcal{E}$-resolution) need not be a decidable set of inferences.

Δ is $\underline{\mathcal{E}\text{-complete}}$ provided that if $\forall(\mathcal{C}) \models_{\mathcal{E}} 0$ then $\Delta(\mathcal{C} \cup \mathcal{E} \cup [x=x])$ contains a refutation, where

$$\Delta(\mathcal{B}) =_{df} \{\underline{\underline{\mathcal{D}}} \in \Delta: \text{Base}(\underline{\underline{\mathcal{D}}}) \leq \mathcal{B}^{\sim}\}$$

Δ is $\underline{\Gamma\text{-complete}}$ provided that if $\mathcal{C} \models_{\Gamma}^{*} 0$ then $\Delta(\mathcal{C} \cup Ax(\Gamma))$ contains a refutation.

Δ is (weakly) $\underline{\Delta'\text{-complete}}$ provided that if $\Delta'(\mathcal{C})$ contains a refutation then so does $\Delta(\mathcal{C})$.

Δ is $\underline{\text{strongly } \Delta'\text{-complete}}$ provided that if $\Delta'(\mathcal{B})$ contains a refutation then so does $\Delta(\mathcal{B}) \cap \Delta'(\mathcal{B})$.

Δ and Δ' are $\underline{\text{compatible}}$ provided that Δ is strongly Δ' complete and Δ' is strongly Δ-complete:

$\underline{\text{Proposition 1}}$. If Δ and Δ' are compatible and either Δ or Δ' is $\mathcal{E}$-complete, then $\Delta \cap \Delta'$ is also $\mathcal{E}$-complete.

$\underline{\Gamma\text{-Closure completeness}}$. There is another completeness concept for refinements which turns out to be quite useful in their formal analysis. Given a Γ-refinement Δ, define Δ^{-}, the $\underline{\text{closure}}$ of Δ by

$$\Delta^{-} =_{df} \{\underline{\underline{\mathcal{D}}}: \underline{\underline{\mathcal{D}}} \text{ is a (possibly infinite) deduction such that for each finite subdeduction } \underline{\underline{\mathcal{D}}}' \text{ of } \underline{\underline{\mathcal{D}}} \text{ where } \text{Base}(\underline{\underline{\mathcal{D}}}') \leq \text{Base}(\underline{\underline{\mathcal{D}}}),\ \underline{\underline{\mathcal{D}}}' \in \Delta\}$$

A deduction $\underline{\underline{\mathcal{D}}}$ is $\underline{\text{complete}}$ in $\Delta^{-}(\mathcal{B})$ provided that either $0 \in \underline{\underline{\mathcal{D}}}$ or $\underline{\underline{\mathcal{D}}}$ is a maximal member of $\Delta^{-}(\mathcal{B})$ (with respect to the subdeduction relation).

$\underline{\text{Definition 2}}$. A Γ-refinement Δ is $\underline{\Gamma\text{-closure complete}}$ provided that if $\mathcal{B}$ is Γ-refutable then each complete deduction in

$\Delta^{-}(\mathcal{B} \cup Ax(\Gamma))$ contains $0 (\mathcal{B} \leq \mathcal{C}_V)$.

Proposition 3. Suppose that Γ is $\mathcal{E}$-adequate and Δ is Γ-closure complete. Then $\mathcal{B}$ is $\mathcal{E}$-consistent iff some complete deduction in $\Delta^{-}(\mathcal{B} \cup Ax(\Gamma))$ fails to contain 0.

The proof is straightforward.

1.3.5 Refutation Procedures

Intuitively, we think of a refutation procedure Π for a Γ-refinement Δ as a procedure which, when given a (finite) clause-set $\mathcal{B}$, iteratively computes a sequence $(\underline{\underline{\mathcal{D}}}_k : k \in N)$ of finite deductions in $\Delta(\mathcal{B} \cup Ax(\Gamma))$ such that

(i) $\underline{\underline{\mathcal{D}}}_k$ is a subdeduction of $\underline{\underline{\mathcal{D}}}_{k+1}$ $(k \in N)$;

(ii) if $\underline{\underline{\mathcal{D}}}_{j+1} = \underline{\underline{\mathcal{D}}}_j$ then $\underline{\underline{\mathcal{D}}}_{j+k} = \underline{\underline{\mathcal{D}}}_j$ $(k \in N)$; and

(iii) if $0 \in \mathcal{D}_j$ then $\underline{\underline{\mathcal{D}}}_{j+1} = \underline{\underline{\mathcal{D}}}_j$.

If $\underline{\underline{\mathcal{D}}}_{j+1} = \underline{\underline{\mathcal{D}}}_j$ for some j then we say that $\Pi(\mathcal{C})$ terminates, and we set $\Pi(\mathcal{C}) = \underline{\underline{\mathcal{D}}}_j$ where $\underline{\underline{\mathcal{D}}}_{j+1} = \underline{\underline{\mathcal{D}}}_j$. Otherwise, we define $\Pi(\mathcal{C})$ to be the limit of $(\underline{\underline{\mathcal{D}}}_i : i \in N)$, which is the deduction $\underline{\underline{\mathcal{D}}} = (\mathcal{D}, \prec)$ in $\Delta^{-}(\mathcal{C} \cup Ax(\Gamma))$ (§1.3.4) defined by

$$\mathcal{D} =_{df} \cup \{\mathcal{D}_i : i \in N\} \quad ; \text{ and}$$

$$\prec =_{df} \cup \{\prec_i : i \in N\} \quad .$$

The following definition is motivated by this intuitive concept:

Definition 4. A refutation procedure for (Γ, Δ) is a total function Γ from finite clause-sets to Γ-deductions in Δ such that

(i) $\Pi(\mathcal{B}) \in \Delta^{-}(\mathcal{B} \cup Ax(\Gamma))$;

(ii) if $0 \in \Pi(\mathcal{B})$ then $\Pi(\mathcal{B})$ is finite ;

(iii) $\{(\mathcal{B}, \Pi(\mathcal{B})) : \Pi(\mathcal{B}) \in \Delta\}$ is computable.

Π is <u>complete</u> (for (Γ,Δ)) provided that, in addition to (i)-(iii),

(iv) $\Pi(\mathcal{B})$ is a complete deduction in $\Delta^{-}(\mathcal{B} \cup Ax(\Gamma))$.

Π is <u>$\mathcal{E}$-complete</u> provided that, in addition to (i)-(iii),

(v) If $\mathcal{B}$ is $\mathcal{E}$-inconsistent then $0 \in \Pi(\mathcal{B})$.

<u>Proposition 5</u>. Suppose that Π is a complete refutation procedure for (Γ,Δ), where Γ is an $\mathcal{E}$-adequate calculus and Δ is Γ-closure complete. Then $0 \in \Pi(\mathcal{B})$ iff $\forall(\mathcal{B}) \models_{\mathcal{E}} 0$, and Π may be viewed as a partial decision procedure for $\mathcal{E}$-inconsistency on the basis of the following classification of $\Pi(\mathcal{B})$:

<u>Decided</u>: $\Pi(\mathcal{B})$ is finite, whence <u>$\mathcal{B}$ is $\mathcal{E}$-inconsistent iff $0 \in \Pi(\mathcal{B})$</u>.

<u>Undecided</u>: $\Pi(\mathcal{B})$ is infinite, whence <u>$\mathcal{B}$ is $\mathcal{E}$-consistent</u>.

<u>Proof</u>. Suppose $0 \in \Pi(\mathcal{B})$. Then $\forall(\mathcal{B}) \models_{\mathcal{E}} 0$ by $\mathcal{E}$-soundness of Γ and the fact that $\Pi(\mathcal{B})$ is a Γ-deduction from $\mathcal{B}^{\sim}$, and $\Pi(\mathcal{B})$ is finite by definition of Π.

Now suppose $0 \notin \Pi(\mathcal{B})$. Then $\mathcal{B}$ is not Γ-refutable, because $\Pi(\mathcal{B})$ is a complete deduction in $\Delta^{-}(\mathcal{B} \cup Ax(\Gamma))$. Thus, $\mathcal{B}$ is $\mathcal{E}$-consistent by $\mathcal{E}$-completeness of Γ. If $\Pi(\mathcal{B})$ is <u>finite</u> then $\mathcal{E}$-consistency of $\mathcal{B}$ has been verified "effectively" rather than "in the limit". ∎

<u>Remark</u>. Suppose that Γ is $\mathcal{E}$-adequate and Π is a refutation procedure for (Γ,Δ). Then in the absence of further conditions on Π

and Δ , we can only use Π as a partial verification procedure for $\mathcal{E}$-inconsistency: if $0 \in \Pi(\mathcal{B})$ then $\Pi(\mathcal{B})$ is finite and $\mathcal{B}$ is $\mathcal{E}$-inconsistent; otherwise $\mathcal{B}$ may or may not be $\mathcal{E}$-inconsistent.

1.3.6 A Complete Refutation Procedure

Consider the intuitive model of a refutation procedure in (Γ,Δ) used to motivate Definition 4 in §1.3.5. Evidently, the behavior of Π in computing $(\underline{\underline{\mathcal{D}}}_i : i \in N)$ is a function of three more or less independent parameters (in addition to the set of premises):

(α) a set $\mathcal{E}$ of axioms ($\mathcal{E} = Ax(\Gamma)$);

(β) a Γ-refinement Δ ;

(γ) a search strategy, which selects the inferences to be generated next in obtaining $\underline{\underline{\mathcal{D}}}_{k+1}$ from $\underline{\underline{\mathcal{D}}}_k$.

The proof procedure Ref below represents (β) by a "resolving function" and (γ) by an "enqueuing function".

A resolving function for (Γ,Δ) (or for Δ) is a function Res which, given a clause A and a deduction $\underline{\underline{\mathcal{D}}}$ in Δ , computes a set $Res(A,\underline{\underline{\mathcal{D}}})$ where

$$Res(A,\underline{\underline{\mathcal{D}}})^{\sim} = \{C: \mathcal{B} \cup \{A\} \vdash_{\Gamma} C \text{ where } \mathcal{B} \subseteq \mathcal{D} \text{ , and the deduction } \underline{\underline{\mathcal{D}}}' \text{ , obtained by adding } (\mathcal{B} \cup \{A\} \vdash_{\Gamma} C) \text{ to } \underline{\underline{\mathcal{D}}} \text{ , is in } \Delta\} .$$

Notation. Res_{Δ} will denote the (essentially unique) resolving function for Δ . (Γ is used only to simplify the description of Res above: Res_{Δ} is essentially determined by Δ .)

Δ is finitary provided that $Res(A,\underline{\underline{\mathcal{D}}})$ is finite ($\underline{\underline{\mathcal{D}}} \in \Delta$, $A \in \mathcal{C}_V$).

Enqueuing Functions. A queue over $\mathcal{C}_V$ is a pair (Q,Enq) where Q is a (finite or infinite) sequence in $\mathcal{C}_V$ and Enq is an "enqueuing function" (over $\mathcal{C}_V$). Let $\mathrm{Seq}(\mathcal{C}_V)$ be the class of all sequences (finite or infinite) of clauses ($\mathcal{C}_V$), and let $\mathrm{Set}(\mathcal{C}_V)$ be the class of all finite sets of clauses. An enqueuing function is a computable functional $\mathrm{Enq}: \mathrm{Seq}(\mathcal{C}_V) \times \mathrm{Set}(\mathcal{C}_V) \to \mathrm{Seq}(\mathcal{C}_V)$ satisfying (i) and (ii) for each pair (Q,T) in $\mathrm{Seq}(\mathcal{C}_V) \times \mathrm{Set}(\mathcal{C}_V)$:

(i) $\{\mathrm{Enq}(Q,T)(k): k \in \mathrm{Domain}(\mathrm{Enq}(Q,T))\} = \{Q(k): k \in \mathrm{Domain}(Q)\} \cup T$.

(ii) If $Q(k) \neq C$ $(k \in \mathrm{Domain}(Q))$ and $\mathrm{Enq}(Q,T)(i) = \mathrm{Enq}(Q,T)(j)$ then $i=j$.

We say that Enq is fair, or that Enq implements fair scheduling, provided that in addition to (i) and (ii),

(iii) For any sequence Q in $\mathrm{Seq}(\mathcal{C}_V)$, any clause C in Q , and any class $\{T_i: i \in N\} \subseteq \mathrm{Set}(\mathcal{C}_V)$, C is the first clause of some sequence Q_i in $\{Q_k: k \in N\}$, where $Q_0 = Q$ and Q_{k+1} is defined from Q_k by

$$Q_{k+1} =_{df} \mathrm{Enq}(Q_k', T_k)$$

where Q_k' is the result of deleting the first element $(Q_k(0))$ from Q_k .

Remark. If Enq is fair then no element will "remain in the queue (Q) forever" during a computational process which alternately removes the next (first) element of the queue and "enqueues a finite set of (new) elements" by means of Enq .

Bound parameters of Ref consist of $\mathcal{C}$, a constant parameter of type $\mathrm{Set}(\mathcal{C}_V)$. $\mathcal{C}$ is the set of premises from which Ref attempts to derive a Γ-refutation.

Free parameters of Ref consist of $\mathcal{E}$, Δ, and Enq:

$\mathcal{E}$: a set of axioms;

Δ: a finitary (Γ-)refinement (where $\mathrm{Ax}(\Gamma) = \mathcal{E}$).

Enq: a fair enqueuing function over clauses.

State variables of Ref consist of Q, R, A, T:

Q: a variable over $\mathrm{Seq}(\mathcal{C}_V)$;

R: $(R, \prec_R)$, a variable over (finite) deductions in Δ;

A: a variable over $\mathcal{C}_V$; and

T a variable over $\mathrm{Set}(\mathcal{C}_V)$.

Now Ref is defined on $\mathrm{Set}(\mathcal{C}_V)$ by

```
Ref(C) =df [Q: = Enq(Enq(Nil, E), C) ;
            R = (0,0) ;
            Result: If  Q = Nil
                        then  R
                        else  [Next(A,Q) ;
                               Subsume(A,R);
                               If  A = 0
                               then  R
                               else [Res_Δ(A,R,T) ;
                                     Q: = Enq(Q,T) ;
                                     Result ]]]
```

Notes on operation. Evaluation of $\underline{\mathrm{Ref}}(\mathcal{C})$ determines a computation sequence $((Q_i, \underline{\underline{R}}_i): i \in N)$ as follows:

1. Q_0 = a bijective sequence onto

2. $\underline{\underline{R}}_0 = (0,0)$, the empty deduction.

3. If Q_k = Nil then $Q_{k+\ell} = Q_k$ and $\underline{\underline{R}}_{k+\ell} = \underline{\underline{R}}_k$ $(\ell \in N)$; otherwise Q_{k+1} and R_{k+1} are defined in steps 4-9 below.

4. Next (A,Q) sets $\underline{A} = Q_k(0)$, the first element of Q_k, and sets $\underline{Q} = (Q_k(i+1): i \in N)$.

5. Subsume(A,R) normally inserts A in $\underline{R}$, thereby defining $R_{k+1} = R_k \cup \{A\}$. However, this operation may delete A from $\underline{R}$ if A is subsumed by a member of $R_k - \{A\}$, and it may delete clauses of $R_k - \{A\}$ subsumed by A.

6. If A = 0 then a refutation has been found: let $Q_{k+\ell} = Q'_k$, $\underline{\underline{R}}_{k+\ell} = (R_{k+1}, \prec_k)$ $(\ell \in N)$. Otherwise continue with steps 7-9.

7. $\mathrm{Res}_\Delta(\underline{A},\underline{\underline{R}},\underline{T})$ sets $\underline{T} = \mathrm{Res}_\Delta(A,\underline{\underline{R}})$ (as in the definition of resolving function) and extends $\prec_k$ to $\prec_{k+1}$ in order to include members of $T - R_{k+1}$, which will subsequently be selected from $\underline{Q}$.

8. $\underline{Q} := \mathrm{Enq}(Q,T)$ inserts the clauses of T into $\underline{Q}$, defining Q_{k+1}.

9. The recursive conditional expression labeled Result is now evaluated with $(\underline{Q},\underline{\underline{R}}) = (Q_{k+1},\underline{\underline{R}}_{k+1})$, as described in Steps 3-8.

Proposition 6. Suppose Subsume(A,R) simply inserts A into $\underline{R}$. Then Ref is a complete refutation procedure for (Γ,Δ).

Proof. Suppose the contrary: $\underline{\underline{\mathrm{Ref}}}(\mathcal{C})$ is not a complete deduction in $\Delta^-(\mathcal{C} \cup \mathrm{Ax}(\Gamma))$. Then it is easily verified that $0 \notin \mathrm{Ref}(\mathcal{C})$

and $\mathrm{Ref}(\mathcal{C})$ is infinite. Let $\underline{\underline{\mathcal{D}}}'$ be a member of $\Delta^{-}(\mathcal{C} \cup \mathrm{Ax}(\Gamma))$ such that $\underline{\mathrm{Ref}}(\mathcal{C})$ is a proper subdeduction of $\underline{\underline{\mathcal{D}}}' = (\mathcal{D}', \prec')$, whence $\mathcal{D}' - \mathrm{Ref}(\mathcal{C}) \neq 0$. It follows that $\mathcal{D}' - \mathrm{Ref}(\mathcal{C})$ contains a clause C which is minimal with respect to $\prec'$. C is not in $(\mathcal{C} \cup \mathrm{Ax}(\Gamma))^{\sim}$ due to the initialization of Q and the assumption that Enq is fair. Therefore $\mathcal{B} \underset{\underline{\underline{\mathcal{D}}}'}{\vdash} C$, where $\mathcal{B} \subseteq \mathrm{Ref}(\mathcal{C})$ by minimality of C. Let $\mathcal{B} = \{A_j\} \cup \mathcal{B}'$ where A_j is the last clause of $\mathcal{B}$ to be transferred from $\underline{Q}$ into R. We claim that $C \in Q_{j+1}$ --i.e., that $C \in \mathrm{Res}(A_j, R_j')$ (where the transfer of A_j to $\underline{Q}$ occurs when $\underline{Q} = Q_k$). Indeed, $(R_j' \cup \{C\}, \prec')$ is a finite subdeduction of $\underline{\underline{\mathcal{D}}}'$ and is therefore in $\Delta(\mathcal{C} \cup \mathrm{Ax}(\Gamma))$ by the definitions of Δ^{-} and <u>refinement</u>. It follows (by the assumption on Enq) that $C \in \mathrm{Ref}(\mathcal{C})$, a contradiction.

<u>Discussion</u>. The procedure Ref shows how easily the proof-structuring aspects (Δ) of a refutation procedure can be separated from the proof-search aspects (Enq). The fairness constraint on Enq is the mildest constraint one could reasonably expect; it merely excludes the pitfalls of pure depth-first search. It is easily shown that <u>Ref</u> is also <u>irredundant</u> in the sense that it generates each proof-tree in $\Delta(\mathcal{C} \cup \mathcal{E})$ only once. The elimination of redundancy due to <u>subsumption</u> relations among clauses in $\underline{R}$ is easily handled by letting <u>Subsume</u> delete certain subsumed clauses of $R \cup \{A\}$; this is illustrated by a closure-computation procedure for reduction systems in §3. A refutation search illustration in §D is based on a simulation of $\underline{\mathrm{Ref}}(\mathcal{C})$ with appropriate choices of Enq, Δ, <u>Subsume</u>, and $\mathcal{C}$. In general, it can be said that <u>Ref</u> is an appropriate refutation procedure for use with Γ-closure complete refinements (Proposition 5).

1.3.7 Analysis of Resolution-Based Deductions

A resolution-based deduction is one which is generated by a resolution-based calculus (for $\mathcal{E}$), whose inferences are all required to be generalized $\mathcal{E}$-resolution inferences (below). Each calculus investigated in §2 is resolution-based.

There is a small class of basic relations and transformations on resolution-based deductions which is so frequently useful in the design or analysis of refinements that it deserves to be treated as a part of the basic theory of proof procedures rather than as a part of the solution to some particular design problem. Some of these relations and transformations are described below so as to avoid duplication of effort in §2.

A generalized $\mathcal{E}$-resolution inference has the form

$$\{B_i: i < n\} \vdash (B_0 - C_0)\theta \vee \cdots \vee (B_{n-1} - C_{n-1})\theta \vee C_n\theta \qquad (3)$$

where

(i) $C_i \subseteq B_i$ $(i=0,\cdots,n-1)$;

(ii) $\{C_i\theta: i < n\} \models C_n\theta$;

(iii) $\theta \cdot \theta = \theta$; and

(iv) if x does not occur in $\{C_i: i \leq n\}$ then $x\theta = x$.

$\{C_i: i < n\}$ is the kernel of (3), C_n the residual of (3), and θ the induced substitution of (3).

Convention. While the kernel, residual, and induced substitution of (3) are not in general uniquely determined by (3), an appropriate choice function will often be tacitly employed to select a unique kernel, residual, and induced substitution for each generalized $\mathcal{E}$-resolution

inference.

Remarks

1. Each inference (3) is $\mathcal{E}$-sound; for suppose ϕ is an $\mathcal{E}$-interpretation wherein B_i is valid $(i=0,\cdots,n-1)$. Then $\phi(B_i\theta) = 1$ $(i=0,\cdots,n-1)$. Suppose $\phi((B_i - C_i)\theta) = 0$ $(i=0,\cdots,n-1)$. Then $\phi(C_j\theta) = 1$ $(i=0,\cdots,n-1)$, whence $\phi(C_n\theta) = 1$ due to (ii), and the conclusion of (3) is true under ϕ.

2. Generalized $\mathcal{E}$-resolution inferences relativize to $\mathcal{E}$ the generalized resolution principle of Robinson [67].

3. Conditions (iii) and (iv) can be imposed without loss of generality provided that θ is a m.g.s.u. of some collection of sets of terms occurring in $\{C_i: i \leq n\}$, which is normally the case.

Descendants. For each Γ-deduction $\mathcal{D}$, the relation of descendants is the smallest reflexive and transitive relation on occurrences of atoms in clauses of $\mathcal{D}$ such that for each inference (3) in $\mathcal{D}$, each occurrence $(B_i \vee \underline{q},*)$ where $q \in B_i - C_i$ has a descendant $(C \vee \underline{q\theta},*)$ in the conclusion.

Ancestor is the converse of descendant.

The transformations on Γ-deductions described below do not always yield Γ-deductions. However, most of them do provided that Γ satisfies the following:

Assumption 1. Γ is a resolution-based calculus for $\models_{\mathcal{E}}$ such that $\models_{\mathcal{E}} Ax(\Gamma)$ and, for each Γ-inference $(\mathcal{B} \vdash_{\Gamma} C)$ with kernel $\{C_i: i< n\}$, residual C_n, and induced substitution θ, conditions (v)-(viii) hold:

(v) $\mathcal{B}\eta \vdash_\Gamma C\eta$ for each invertible substitution η, and this inference has kernel $\{C_i\eta: i<n\}$, residual $C_n\eta$, and induced substitution $\eta^{-1}\cdot\theta\cdot\eta$.

(vi) $\{C_i: i<n\} \vdash_\Gamma C_n\theta$, and this inference has kernel $\{C_i: i<n\}$, residual C_n, and induced substitution θ.

(vii) $\{A_i \vee C_i: i<n\} \vdash_\Gamma (A_0 \vee \cdots \vee A_n)\theta \vee C_n\theta$.

(viii) If θ divides τ then $\{C_i\tau: i<n\} \vdash_\Gamma C_n\tau$.

Remark. If Γ satisfies Assumption 1 then Γ is generated from the set of "kernel" Γ-inferences (vi). Each resolution-based calculus defined in §2 is representable as a refinement of a calculus satisfying Assumption 1. In future developments it may be useful to incorporate this assumption into the definition of resolution-based so as to avoid overlapping functions of calculi and refinements in research on proof procedures.

A unit deduction is a deduction $\underline{\underline{\mathcal{D}}}$ such that $\mathcal{D} \subseteq \mathcal{L}_V$. The embedding transformation defined below is useful for defining refinements in terms of classes of unit Γ-deductions.

Embeddings. The embedding of $(\{B_i: i<n\} \vdash_\Gamma C)$ (satisfying (i)-(iv)) in $\{A_i \vee B_i: i<n\}$ is the ($\mathcal{E}$-sound) inference $(\{A_i \vee B_i: i<n\} \vdash ((A_0 \vee B_0) - C_0)\theta \vee \cdots \vee ((A_{n-1} \vee B_{n-1}) - C_{n-1})\theta \vee C_n\theta$. Whether or not this is a Γ-inference depends upon the choice of $A_0,\cdots,A_{n-1}$. (In any case, the embedding of the "kernel inference" (vi) in $\{B_i: i<n\}$ is the Γ-inference $\{(B_i: i<n\} \vdash_\Gamma C)$. Let $\underline{\underline{\mathcal{D}}}'$ be a Γ-deduction from $\{B_i: i<n\}$, and let $\mathcal{B} = \{A_i \vee B_i: i<n\}$. The embedding of $\underline{\underline{\mathcal{D}}}'$ in $\mathcal{B}$ is defined

on the basis of the Γ-inference embedding by induction on the number of inferences in $\underline{\underline{\mathcal{D}}}'$. If $\underline{\underline{\mathcal{D}}}'$ contains no inferences then the embedding is just $(\mathcal{B},0)$. Suppose $\underline{\underline{\mathcal{D}}}'$ contains an initial inference $(\mathcal{B}_1' \vdash_\Gamma C_1')$. Let $(\mathcal{B}_1 \vdash C_1)$ be the embedding of $(\mathcal{B}_1' \vdash_\Gamma C_1')$ in $\mathcal{B}$ (or in $\{A_i \vee B_i : B_i \in \mathcal{B}_i'\}$), and let $\underline{\underline{\mathcal{D}}}_1'$ be the result of deleting $(\mathcal{B}_1' \vdash_\Gamma C_1')$ from $\underline{\underline{\mathcal{D}}}'$ (so that $C_1' \in \text{Base}(\underline{\underline{\mathcal{D}}}_1')$). Let $\mathcal{B}_1 = \mathcal{B} \cup \{C_1\}$, and complete the embedding $\underline{\underline{\mathcal{D}}}_1$ of $\underline{\underline{\mathcal{D}}}_1'$ in $\mathcal{B}_1$. Let $\underline{\underline{\mathcal{D}}}$ be the result of prefixing the embedded inference $(\mathcal{B}_1 \vdash C_1)$ to $\underline{\underline{\mathcal{D}}}_1$. $\underline{\underline{\mathcal{D}}}$ is the <u>embedding</u> of $\underline{\underline{\mathcal{D}}}'$ in $\mathcal{B}$. Again, whether or not $\underline{\underline{\mathcal{D}}}$ is a Γ-deduction depends upon the choice of $(A_i : i < n)$.

<u>Ground deductions</u>. A <u>ground</u> Γ-inference is one whose induced substitution is ε, the identity substitution. A <u>ground</u> Γ-deduction is one whose constituent Γ-inferences are all ground Γ-inferences. If $\underline{\underline{\mathcal{D}}}(C)$ is a ground Γ-deduction based on $\mathcal{B}$, then $\mathcal{B} \models_{\varepsilon} C$.

<u>General deductions</u>. A Γ-deduction $\underline{\underline{\mathcal{D}}}$ is <u>general</u> provided that it satisfies (i) and (ii):

(i) If B and C are two clauses of $\underline{\underline{\mathcal{D}}}$ which share a variable, then either $B \in \mathcal{D}(C)$ or $C \in \mathcal{D}(B)$. (Thus, if $\mathcal{B} \vdash_{\underline{\underline{\mathcal{D}}}} C$ then $\mathcal{B}$ is separated.)

(ii) If $(\mathcal{B}_i \vdash_{\underline{\underline{\mathcal{D}}}} C_i)$ has induced substitution $\theta_i (i=1,2)$ and $C_1 \neq C_2$ then either $x\theta_1 = x$ or $x\theta_2 = x$ $(x \in V_I)$.

<u>Composite substitutions</u>. Let $\underline{\underline{\mathcal{D}}}$ be a <u>general</u> Γ-deduction, and let $(\theta_i : i < n)$ be a list of all the induced substitutions ordered so that if θ_i is induced by $(\mathcal{B}_i \vdash_{\underline{\underline{\mathcal{D}}}} C_i)$ and θ_j is induced by $\mathcal{B}_j \vdash_{\underline{\underline{\mathcal{D}}}} C_j$ where $C_i \in \mathcal{D}(C_j)$ then $i \leq j$. The <u>composite substitution</u>

$\sigma_{\underline{\mathcal{D}}}$ (induced by $\underline{\mathcal{D}}$) is defined by

$$\sigma_{\underline{\mathcal{D}}} =_{df} (\theta_0 \cdot \cdots \cdot \theta_{n-1})$$

Proposition 7. $\sigma_{\underline{\mathcal{D}}}$ is well defined, and θ_k divides $\sigma_{\underline{\mathcal{D}}}$ $(k=0,\cdots,n-1)$.

The proof is based on the simple observation that if $i \neq j$ then either $x\theta_i = x$ or $x\theta_j = x$ (because $\underline{\mathcal{D}}$ is general).

Instantiation of Γ-deductions. Let $\underline{\mathcal{D}}$ be a ground or general Γ-deduction $(\mathcal{D},\prec)$, and suppose that the induced substitution for each inference in $\underline{\mathcal{D}}$ divides σ. Then $\underline{\mathcal{D}}\sigma$, the instantiation of $\underline{\mathcal{D}}$ by σ, is the deduction $(\mathcal{D}\sigma,\prec_\sigma)$ where $A\sigma \prec_\sigma B\sigma$ iff $A \prec B$ and $A\sigma \neq B\sigma$. The following property of instantiation is easily verified:

Proposition 8. If $\underline{\mathcal{D}}$ is a general Γ-deduction where Γ satisfies Assumption 1, then $\underline{\mathcal{D}}\sigma_{\underline{\mathcal{D}}}$ is a ground Γ-deduction.

1.3.8 Liftable Calculi and Refinements

The branches in a deduction $\underline{\mathcal{D}} = (\mathcal{D},\prec)$ are the maximal chains in $(\prec \cap (\mathcal{D} \times \mathcal{D}))$. A branch mapping (for $\underline{\mathcal{D}}$, or for $(\underline{\mathcal{D}}, \mathcal{A})$) is a mapping π from the branches of $\underline{\mathcal{D}}$ into a clause-set $\mathcal{A}$ such that

(i) $\pi(\mathcal{B})$ subsumes the initial ($\prec$-minimal) element of $\mathcal{B}$; and

(ii) if $\pi(\mathcal{B}_1)$ shares any variables with $\pi(\mathcal{B}_2)$ then $\mathcal{B}_1 = \mathcal{B}_2$.

Thus, the image of π in $\mathcal{A}$ is a separated set of clauses.

Let Γ be a resolution-based calculus for $\mathcal{E}$. Γ is liftable provided that if $\{B_i': i<n\} \vdash_\Gamma C'$ and $\{B_i: i<n\}$ is a separated clause-set such that B_i subsumes B_i' $(i=0,\cdots,n-1)$, either B_j subsumes C' where $j<n$, or $\{B_i: i<n\} \vdash_\Gamma C$ where C subsumes C'.

Γ-liftings. Let Γ be a liftable calculus, and let $\underline{\underline{\mathcal{D}}}'$ be a Γ-deduction. Let π be a branch mapping for $\underline{\underline{\mathcal{D}}}'$. A Γ-lifting $\underline{\underline{\mathcal{D}}}$ of $\underline{\underline{\mathcal{D}}}'$ is obtained from $(\underline{\underline{\mathcal{D}}}',\pi)$ by induction on the number of inferences in $\underline{\underline{\mathcal{D}}}'$, as follows.

Suppose $\underline{\underline{\mathcal{D}}}'$ contains no inferences. Then the trivial deduction $(\{\pi\{B'\}\colon B' \in \text{Base}(\underline{\underline{\mathcal{D}}}'),\prec)$ is a Γ-lifting of $\underline{\underline{\mathcal{D}}}'$

Suppose $\mathcal{A}' \vdash_{\underline{\underline{\mathcal{D}}}'} C'$ where $\mathcal{A}' = \{A_i'\colon i<n\}$. Let $\underline{\underline{\mathcal{D}}}_1'$ be the result of deleting this inference inference. Let $\{\mathcal{B}_i\colon i<\ell\}$ be the set of branches in $\underline{\underline{\mathcal{D}}}'$ which contain C', and let $\{\pi(\mathcal{B}_i)\colon i<n\} = \{A_i\colon i<n\}$ where A_i subsumes A_i' and $\pi(\mathcal{B}_j) = A_i$ for some j such that $A_i' \in \mathcal{B}_j \cup \text{Base}(\underline{\underline{\mathcal{D}}}')$ $(i=0,\cdots,n-1)$.

Case 1: A_j subsumes C' where $j<n$. Define π_1 on the branches of $\underline{\underline{\mathcal{D}}}_1'$ by

$$\pi_1(\mathcal{B}) =_{df} \begin{cases} A_j & \text{, if } C' \in \mathcal{B} \text{ ;} \\ \pi(\mathcal{B}) & \text{, otherwise.} \end{cases}$$

Let $\underline{\underline{\mathcal{D}}}_1$ be a Γ-lifting of $\underline{\underline{\mathcal{D}}}_1'$ (using π_1), and let $\underline{\underline{\mathcal{D}}} = \underline{\underline{\mathcal{D}}}_1$.

Case 2: A_i does not subsume C' $(i=0,\cdots,n-1)$. It follows by liftability of Γ that $\{A_i\colon i<n\} \vdash_\Gamma C$ where C subsumes C'. Suppose without loss of generality that C shares no variables with other clauses of $\mathcal{A}$, and let $\mathcal{A}_1 = \mathcal{A} \cup \{C\}$. Define π_1 on branches of $\underline{\underline{\mathcal{D}}}_1'$ by

$$\pi_1(\mathcal{B}) = \begin{cases} C & \text{, if } C' \in \mathcal{B} \text{ ;} \\ \pi(\mathcal{B}) & \text{, otherwise .} \end{cases}$$

Let $\mathfrak{D}_1$ be a π_1-lifting of $\mathfrak{D}_1'$ (using induction), and let $\mathfrak{D}$ be the result of prefixing $(\{A_i : i < n\} \mid_\Gamma C)$ to $\mathfrak{D}$.

Remark. While the induced substitutions in the inferences of $\mathfrak{D}$ are not mentioned explicitly, we may assume without loss of generality (on the basis of §1.3.7) that $\mathfrak{D}$ is general. (Property (ii) of π justifies this assumption.) Thus, we have the following:

Proposition 9. Suppose that Γ is liftable and $\mathfrak{D}'$ is a Γ-deduction. Let π be a branch-mapping for $(\mathfrak{D}', \mathcal{A})$. Then each Γ-lifting $\mathfrak{D}$ of $\mathfrak{D}'$ obtained from $(\mathfrak{D}', \pi)$ is a general Γ-deduction $\mathfrak{D}$ such that $\text{Base}(\mathfrak{D}) \subseteq \mathcal{A}$ and each conclusion of $\mathfrak{D}'$ is subsumed by a conclusion of $\mathfrak{D}$.

Let Δ be a Γ-refinement where Γ is liftable. Δ is liftable provided that for each ground refutation $\mathfrak{D}'$ in Δ and each branch-mapping π for $\mathfrak{D}'$; some Γ-lifting of $\mathfrak{D}'$ based on $(\mathfrak{D}', \pi)$ also in Δ; Δ is strongly liftable provided that Δ contains each Γ-lifting of $\mathfrak{D}'$ based on $(\mathfrak{D}', \pi)$.

Liftability results for calculi and refinements can be very useful in the derivation of completeness results (§3).

2. PROBLEM SOLUTION

The purpose of this chapter is to define a class of normal refinements which can be designed hierarchically, and to show how this design process relates to the structured programming of specialized proof procedures.

In addition to Γ-refinement (§1.3.4), two related concepts will be useful in the overview which follows. A Γ-macro-refinement is a Γ-refinement Δ_M such that for some "higher level" calculus Γ_M, each refutation in Δ_M has a decomposition into realizations of Γ_M-inferences. A Γ-micro-refinement is a Γ_μ-refinement Δ_μ wherein each refutation has a decomposition into realizations of Γ-inferences. (Γ_μ is a "lower level" calculus than Γ .)

Thus, if $\mathcal{E} \notin \mathcal{L}_V$ then a normal refinement $\Delta_M \cdot \Delta_\mu$ for $\mathcal{E}$ is a composition (§2.4.1) of a resolution macro-refinement Δ_M with a resolution micro-refinement Δ_μ. Resolution is the calculus which admits $\mathcal{E}$-resolution inferences (Abstract, or §2.1.1) where $\mathcal{E} \leq [x=x]^*$.

$\mathcal{E}$-resolution refinements are the topic of §2.1. Hyper-$\mathcal{E}$-resolution is the higher-level calculus for an $\mathcal{E}$-resolution macro-refinement HR($\mathcal{E}$,≻,s) (§2.1.2). A useful lifting transformation on $\mathcal{E}$-resolution deductions is described in §1.3.8 This transformation takes a ground ($\mathcal{E}$-resolution) deduction $\underline{\underline{\mathcal{D}}}'$, and a clause-set $\mathcal{B}$ such that each member of Base($\underline{\underline{\mathcal{D}}}'$) is subsumed by a corresponding member of $\mathcal{B}$, and produces a general deduction $\underline{\underline{\mathcal{D}}}$ such that each conclusion of $\underline{\underline{\mathcal{D}}}'$ is subsumed by a conclusion of $\underline{\underline{\mathcal{D}}}$.

The investigation of resolution micro-refinements begins in §2.2 with the definition of a basic calculus Γ_S based on the following axiom and rules:

(i) Simple reflexivity: [x=x] ;

(ii) Simple factoring (SF), a restricted form of factoring ;

(iii) Replacement (Rp), a restricted form of paramodulation ;

(iv) Cut, a restricted form of binary (or pairwise) resolution.

A normal deduction is a basic deduction $\mathcal{D}$ wherein SF is not applied to conclusions of Rp-inferences; if C is a conclusion of a binary resolution inference in $\mathcal{D}$ then $\mathcal{D}(C)$ is uniquely decomposable into realizations of resolution inferences. Each normal refinement (below) is a subset of the normal deductions.

The sole function of a resolution micro-refinement Δ_μ is to specify, for each resolution inference ( ), a set of admissible Γ_μ-realizations. In the case of normal refinements, $\Gamma_\mu = \Gamma_S$ and Δ_μ is defined from the unit Γ_S-deductions in Δ_μ by means of a normal embedding transformation (§2.2.3).

The class ND($\mathcal{E}$,$\succ$) of $\mathcal{E}$-normal deductions (§2.3.8) exemplifies the above remarks on Δ_μ . Each unit $\mathcal{E}$-normal deduction has a decomposition into $\mathcal{E}$-normal reductions and highly restricted Rp or Cut inferences. An $\mathcal{E}$-normal reduction reduces a literal p to a literal q by a chain of (ground) Rp-inferences based on equations ($s\theta = t\theta$) in $\mathcal{E}^*$ such that $s\theta \succ t\theta$. The restrictions on Rp and Cut are essentially that the (unit) premises must be irreducible with respect to ($\mathcal{E}$,$\succ$).

Subsequent completeness results (§3.2) require that $\succ$ be an $\mathcal{E}'$-complexity ordering for some set $\mathcal{E}'$ of equations in $\mathcal{E}$ --i.e., $\succ$ is a decidable monotone (§2.3.2) partial order on $\mathcal{T}_V$ which is preserved under certain "$\mathcal{E}'$-normal" substitutions and which well-orders constant terms.

Normal compositions of refinements are defined in §2.4.1. Given an $\mathcal{E}$-resolution refinement Δ and an $\mathcal{E}'$-resolution refinement Δ' where $\mathcal{L}_V \not\supseteq \mathcal{E} \supset \mathcal{E}'$, $\Delta \cdot \Delta'$ is an $\mathcal{E}'$-resolution refinement wherein each refutation has a decomposition $\{\mathcal{D}_i(C_i): i < n\}$ such that $\mathcal{D}_i(C_i)$ realizes an $\mathcal{E}$-resolution inference $(\mathcal{B}_i \vdash C_i)$ $(i \leq n)$, $\{\mathcal{B}_i \vdash C_i: i \leq n\}$ defines a refutation in Δ, and $\mathcal{D}_i(C_i)$ is obtained by embedding (§1.3.7) a unit $\mathcal{E}'$-resolution refutation which is in Δ' $(i \leq n)$. If Δ_M is a resolution refinement (§2.1.1) and Δ_μ is a normal resolution micro-refinement, then $\Delta_M \cdot \Delta_\mu$ is defined similarly.

A normal refinement (for $\mathcal{E}$) is an $\mathcal{E}$-resolution micro-refinement $\Delta_M \cdot \Delta_\mu$ where Δ_M is a resolution refinement and Δ_μ is a resolution micro-refinement. Normally $\Delta_M = (\cdots(\Delta_n \cdot \Delta_{n-1})\cdots\Delta_0)$ where Δ_k is an $\mathcal{E}_k$-resolution refinement, $\mathcal{E} \supseteq \mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ and $\mathcal{E}_0 \subseteq \mathcal{L}_V$.

A normal refutation procedure (for $\models_{\mathcal{E}}$) is one whose search space is a normal refinement (for $\mathcal{E}$). The clause representation and Level function in §2.4.2, based on an analysis of ancestors of atoms occurring in members of $\Delta_M \cdot \Delta_\mu$, makes it clear that $\Delta_M \cdot \Delta_\mu$ can be efficiently incorporated into the design of a normal refutation procedure for $\models_{\mathcal{E}}$. Moreover, an intuitive description of a search strategy for use with $\Delta_M \cdot \Delta_\mu$ is given in terms of search strategies $E_n, \cdots, E_0, E_\mu$, under the assumption that E_K is characterized by a cost function on deductions and a heuristic "cost of completion" estimator.

2.1 $\mathcal{E}$-resolution Refinements

2.1.1 $\mathcal{E}$-resolution

An $\mathcal{E}$-resolution inference is a generalized $\mathcal{E}$-resolution inference

$$\{B_i \vee q_i : i < n\} \vdash (B_0 - C_0)\theta \vee \cdots \vee (B_{n-1} - C_{n-1})\theta \qquad (1)$$

where

(v) $q_i \in C_i \subseteq B_i \vee q_i$ $(i=0,\cdots,n-1)$;

(vi) $C_i\theta = \{q_i\theta\}$ $(i=0,\cdots,n-1)$; and

(vii) $\{q_i\theta : i < n\}$ is $\mathcal{E}$-contradictory.

Thus, the kernel of (1) is $\{C_i : i < n\}$, and the residual of (1) is 0 (§1.3.7). Equivalently, we can define an $\mathcal{E}$-resolution inference to be an inference

$$\{B_i \vee q_i : i < n\} \vdash C \qquad (2)$$

where $(B_0 - q_0)\theta \vee \cdots \vee (B_{n-1} - q_{n-1})\theta \supseteq C \supseteq (B_0\theta - q_0\theta) \vee \cdots \vee (B_{n-1}\theta - q_{n-1}\theta)$, $\{q_i\theta : i < n\}$ is $\mathcal{E}$-contradictory, and θ satisfies (iii) and (iv) in §1.3.7 with $C_i = C \cap (B_i - q_i)\theta$ $(i=0,\cdots,n-1)$. The conclusion C is an $\mathcal{E}$-resolvent of any set which includes the premises.

An $\mathcal{E}$-resolution inference (2) is $\mathcal{E}$-pure provided that it satisfies (i)-(iv):

(i) $B_i \vee q_i$ is not subsumed by any clause in $\mathcal{B} \cup [x=x]$.

(ii) $B_i \vee q_i$ shares no variables with $B_j \vee q_j$ $(0 \leq i < j < n)$.

(iii) If $(B_i \vee \underline{q}_i)$ subsumes $(B_j \vee \underline{q}_j)$ then $(B_i \vee \underline{q}_i)\eta$ and $(B_j \vee q_j)$ are variants (i.e., $(B_i \vee \underline{q}_i)\eta = B_j \vee \underline{q}_j$ where η is invertible) $(0 \leq i < j < n)$.

(iv) C is not a tautology and is not subsumed by any clause in $\{B_i \vee q_i : i < n\} \cup [x=x]$.

$\mathcal{E}$-resolution refers either to the calculus which consists of all $\mathcal{E}$-resolution inferences (no axioms), or else to the class of all $\mathcal{E}$-resolution deductions. Thus, an $\mathcal{E}$-resolution refinement is defined as in §1.3. (See Remark 2 below, and Footnote 1 in §1.3).

Convention. If $\mathcal{E} \subseteq [x=x]^*$ then we drop the prefix $\mathcal{E}$- .

Remarks.

1. $\mathcal{E}$-resolution is not in general decidable. (If $\mathcal{E}$ axiomatizes Group Theory and V_F^0 is infinite, then $\mathcal{E}$-resolution is undecidable by the undecidability of the general word problem for finite extensions of the group theory axioms (§C,[71]).

2. $\mathcal{E}$-resolution inferences realized by proof procedures will normally (or ideally) be $\mathcal{E}$-pure. However, it seems most appropriate to let the refinement determine which of (i)-(iv) are to be satisfied.

2.1.2 Hyper-$\mathcal{E}$-resolution

A renaming is a mapping $r: Q_V \to \mathcal{L}_V$ such that

(i) $r(p) \in \{p, \tilde{p}\}$;

(ii) $r([s=t]) = [s=t]$; and

(iii) $r(p\theta) = r(p)\theta$

r is extended to clauses by

(iv) $r(\sim p) =_{df} \sim r(p)$; and

(v) $r(q_1 \vee \cdots \vee q_n) =_{df} r(q_1) \vee \cdots \vee r(q_n)$.

If $r(C)$ is a positive (negative) clause then C is r-positive (r-negative).

A (negative literal) selection function (for a renaming r) is a mapping $s: \mathcal{C}_V \to \mathcal{C}_V$ such that

$$s(C) = \begin{cases} 0 \text{ , if } C \text{ is r-positive or empty ;} \\ \text{some r-negative literal } q \text{ in } C \text{ , otherwise.} \end{cases}$$

Evidently, r is definable as a function of s :

$$r_s(p) =_{df} \begin{cases} p & \text{, if } s(p) = 0 \\ \tilde{p} & \text{, if } s(p) = p \text{ .} \end{cases}$$

Let $\mathcal{E} = (\mathcal{E}, \succ, s)$ where $\succ$ is a substitutive partial ordering on $\mathcal{A}_V$ wherein equations precede all other atoms and s is a selection function.

A hyper-$\mathcal{E}$-resolution inference is an inference

$$\{B_0 \vee p_0, \cdots, B_{m-1} \vee p_{m-1}, B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}\} \vdash C \qquad (3)$$

satisfying (i)-(vii):

(i) $C = (B_0\theta - p_0\theta) \vee \cdots \vee (B_{m-1}\theta - p_{m-1}\theta) \vee (B_m\theta - \{\tilde{q}_0\theta, \cdots, \tilde{q}_{n-1}\theta\})$.

(ii) $\{p_0\theta, \cdots, p_{m-1}\theta, (\tilde{q}_0\theta \vee \cdots \vee \tilde{q}_{n-1}\theta)\}$ is an $\mathcal{E}$-contradiction.

(iii) $B_i \vee p_i$ is r_s-positive and $r_s(p)$ is a maximal atom in $r_s(B_i \vee p_i)$ with respect to $\succ$ $(i=0,\cdots,m-1)$.

(iv) B_m is r_s-positive or empty and q_i is r_s-positive $(i=0,\cdots, n-1)$.

(v) $B_i \vee p_i$ is not subsumed by a member of $\mathcal{E} \cup [x=x]$ $(i=0,\cdots, m-1)$.

(vi) If $B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$ is subsumed by a member of $\mathcal{E} \cup [x=x]$ then $B_m = 0$ and $(\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}) \in \mathcal{E}^*$.

(vii) If $(B_i \vee q_i)$ subsumes $(B_j \vee q_j)$ then $(B_i \vee q_i)$ and $(B_j \vee q_j)$ are variants.

From (ii) and (iii) it follows that the conclusion C (a hyper-$\mathcal{E}$-resolvent of the premises) is r_s-positive or empty. The major premise of (3) is $B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$; $B_i \vee p_i$ is a minor premise of (3).

Hyper-$\mathcal{E}$-resolution refers either to the calculus which consists of all hyper-$\mathcal{E}$-resolution inferences (no axioms), or else to the class of all hyper-$\mathcal{E}$-resolution deductions.

Convention. If $\mathcal{E} \subseteq [x=x]^*$, $\succ$ is the trivial ordering, and r_s is the identity renaming then the infix "-$\mathcal{E}$-" is replaced by "-E-". Thus, a hyper-E-resolution inference has the form

$$\{B_0 \vee p_0, \cdots, B_{m-1} \vee p_{n-1}, B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_n\} \vdash C$$

where $B_i \vee p_i$ is positive $(i=0,\cdots,m-1)$, B_{m-1} is positive or empty, $(\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1})$ is negative, and $\{p_0\theta, \cdots, p_{m-1}\theta, \tilde{q}_j\theta\}$ is a contradiction $(j=0,\cdots,n-1)$.

An $\mathcal{E}$-resolution inference is an $\mathcal{E}$-resolution inference

$$\{B_0 \vee p_0, \cdots, B_{m-1} \vee p_{n-1}, B_m \vee \tilde{q}_m\} \vdash C \qquad (4)$$

satisfying (i)-(vi):

(i) $C = (B_0\theta - q_0\theta) \vee \cdots \vee (B_{m-1}\theta - q_{m-1}\theta)$

(ii) $\{p_0\theta, \cdots, p_{m-1}\theta, \theta, \tilde{q}_m\theta\}$ is an $\mathcal{E}$-contradiction.

(iii) $B_i \vee p_i$ is r_s-positive and $r_s(p_i)$ is a maximal atom in $r_s(B_i \vee \underline{p}_i)$ with respect to $\succ$ $(i=0,\cdots,m-1)$.

(iv) q_m is r_s-positive.

(v) $B_i \vee \underline{p}_i$ is not subsumed by a member of $\mathcal{E} \cup [x=x]$ $(i=0,\cdots, m-1)$.

(vi) If $B_m \vee \underline{\tilde{q}}_m$ is subsumed by a member of $\mathcal{E} \cup [x=x]$ then $B_m \vee \tilde{q}_m$ is r_s-negative and $(B_m \vee \tilde{q}_m) \in \mathcal{E}^*$.

The major premise of (4) is $B_m \vee \underline{\tilde{q}}_m$; $B_i \vee \underline{p}_i$ is a minor premise.

$\underline{\underline{\mathcal{E}}}$-resolution refers either to the calculus which consists of all $\underline{\underline{\mathcal{E}}}$-resolution inferences (no axioms), or else to the class of all $\underline{\underline{\mathcal{E}}}$-resolution deductions.

Proposition 1. Let $\underline{\underline{\mathcal{D}}}(C)$ be an $\underline{\underline{\mathcal{E}}}$-resolution deduction where C is r_s-positive or empty. Then $\underline{\underline{\mathcal{D}}}(C)$ has a unique decomposition $\{\underline{\underline{\mathcal{D}}}_i(C_i): i \leq \ell\}$ such that $\underline{\underline{\mathcal{D}}}_j(C_j)$ realizes a hyper-$\underline{\underline{\mathcal{E}}}$-resolution inference $\mathcal{B}_j \vdash C_j$ $(j=0,\cdots,\ell)$.

Remark. This proposition is easily proved by induction on the number κ of r_s-positive or empty clauses in $\mathcal{D}(C)$. The following "converse" to Proposition 1 is also easily verified:

Proposition 2. Each hyper-$\underline{\underline{\mathcal{E}}}$-resolution inference (2) has a unique $\underline{\underline{\mathcal{E}}}$-resolution realization.

Thus, the maximal $\underline{\underline{\mathcal{E}}}$-resolution refinement $HR(\mathcal{E},\succ,s)$, where $HR(\mathcal{E},\succ,s) =_{df} \{\underline{\underline{\mathcal{D}}}: \underline{\underline{\mathcal{D}}}$ is an $\underline{\underline{\mathcal{E}}}$-resolution deduction$\}$, is

also a hyper-$\mathcal{E}$-resolution micro-refinement (§2.0).

2.2 A Basic Calculus

The following subsections define a basic calculus Γ_S over $\mathcal{C}_V$. Γ_S has one axiom, [x=x] (the Simple reflexivity axiom) and three basic inference rules: Simple Factoring (SF), Replacement (Rp), and Cut. Cut is the binary (or pairwise) restriction of Resolution.

A basic deduction is a Γ_S-deduction.

2.2.1 Unification and Simplest Unifiers

The conclusion of each basic inference is constructed by instantiating certain constituents of the premises. In order to ensure that the conclusion is as general as possible, the concept of a simplest or most general unifier is used to define the instantiating substitution. Given a property P of substitutions, we say that σ is a simplest (or most general) substitution such that $P(\sigma)$ provided that

(i) $P(\sigma)$; and

(ii) if $P(\tau)$ then σ divides τ, in the sense that $\sigma\cdot\theta = \tau$ for some θ.

Now σ is said to unify a finite nonempty set U of terms or formulas provided that $u\sigma = v\sigma$ $(u,v \in U)$. Robinson [63] verifies a unification algorithm which computes a most-general unifier mgu(U) of U, provided that U is unifiable, and otherwise returns nil.

It is easily verified that if σ is any most general unifier for U then

(i) $\sigma \cdot \sigma = \sigma$ (i.e., σ is <u>idempotent</u>); and

(ii) $mgu(U)\theta = \sigma$ for some invertible substitution θ .

A substitution σ <u>simultaneously</u> unifies a collection $\mathcal{U}$ of subsets of $\mathcal{T}_V \cup \mathcal{L}_V$ provided that σ unifies each member of $\mathcal{U}$. It is shown in [69] (and elsewhere) that there exists a total computable function mgsu from finite collections of finite subsets of $\mathcal{T}_V \cup \mathcal{L}_V$ to $\Sigma_V \cup \{\underline{nil}\}$ such that

$$mgsu(\mathcal{U}) = \begin{cases} \text{a most general simultaneous unifier (mgsu) for } \mathcal{U}, \\ \text{provided that } \mathcal{U} \text{ is simultaneously unifiable;} \\ \\ \underline{nil}\ , \text{ otherwise.} \end{cases}$$

It is easily verified that if σ is any most general simultaneous unifier for $\mathcal{U}$ then

(i) $\sigma \cdot \sigma = \sigma$; and

(ii) $mgsu(\mathcal{U})\theta = \sigma$ for some invertible substitution θ .

<u>Notation.</u> Given a predicate <u>P(x)</u> which holds for only a finite subset $\{x_1, \cdots, x_n\}$ of V_I , we define the substitution

$$[F(x)/x:P(X)] =_{df} [F(x_1)/x_1, \cdots, F(x_n)/x_n]$$

Thus the <u>quotient</u> of σ by τ , $[\sigma/\tau]$ is defined by

$$[\tau/\sigma] =_{df} [x\tau/x : x\sigma = x \text{ and } x\tau \neq x] .$$

Lemma 3. Suppose $\sigma\cdot\sigma = \sigma$ and σ divides τ. Then $\sigma\cdot[\tau/\sigma] = \tau$.

Proof. Suppose $\sigma\cdot\theta = \tau$. If $x\sigma = x$ then clearly $x\sigma[\tau/\sigma] = x\tau$. Suppose $x\sigma \neq x$. If y occurs in $x\sigma$ then $y\sigma = y$ because $x\sigma\sigma = x\sigma$, whence $y\theta = (y\sigma)\theta = y\tau = y[\tau/\sigma]$. It follows that if $x\sigma \neq x$ then $(x\sigma)[\tau/\sigma] = (x\sigma)\theta = x\tau$. ∎

The following lemma shows that mgsu($\mathcal{U}$) can be computed by applying mgu (in any order) to members of $\mathcal{U}$.

Lemma 4. If $\sigma_1 = \text{mgsu}(\mathcal{U}_1)$ and $\sigma_2 = \text{mgsu}(\mathcal{U}_2\sigma_1)$ then $\sigma_1\cdot\sigma_2$ is a mgsu of $\mathcal{U}_1 \cup \mathcal{U}_2$.

Proof. Clearly $\sigma_1\cdot\sigma_2$ simultaneously unifies $\mathcal{U}_1 \cup \mathcal{U}_2$. Suppose τ simultaneously unifies $\mathcal{U}_1 \cup \mathcal{U}_2$. Then $\sigma_1\cdot[\tau/\sigma_1] = \tau$, because τ simultaneously unifies $\mathcal{U}_1$. Therefore $[\tau/\sigma_1]$ simultaneously unifies $\mathcal{U}_2\sigma_1$, whence $\sigma_2\cdot[[\tau/\sigma_1]/\sigma_2] = [\tau/\sigma_1]$ by definition of σ_2. Since $(\sigma_1\cdot\sigma_2)\cdot[[\tau/\sigma_1]/\sigma_2] = \tau$, it follows that $\sigma_1\cdot\sigma_2$ is a mgsu of $\mathcal{U}_1 \cup \mathcal{U}_2$. ∎

2.2.2 Basic Inferences

Basic inferences consist of SF, Rp, and Cut inferences.

A Simple Factoring (SF) inference has the form

$$\{A \vee p\} \vdash A\theta \vee \underline{p\theta} \tag{5}$$

where θ is a m.g.u. of a set $\{p_1,\cdots,p_n\}$ of literals in A which contains p. $A\theta \vee \underline{p\theta}$ is said to be a simple factor of $A \vee p$ on p.

Remark. Notice that the designated literal of the premise in (5) is irrelevant, and that $p\theta$, the designated literal of the conclusion, is the "image" of $\{p_1,\cdots,p_n\}$ under θ . In cases where $\theta = \varepsilon$, so that $A\theta \vee \underline{p\theta} = A \vee \underline{p}$, it may be useful in the definition of a refinement to distinguish $A \vee p$ and $A \vee \underline{p}$ if $A \vee p$ has a different designated literal than p . (See §1.2.3, Convention).

A replacement (Rp) inference has the form

$$\{A \vee \underline{[s=t]},\ B \vee \underline{q[r]}\} \vdash C \vee \underline{q[t]\theta} \qquad (6)$$

where

(i) $C = (A - [s=t])\theta \vee (B - q[r])\theta$; and

(ii) θ is a m.g.u. of $\{r,s\}$.

(6) is general provided that $A \vee \underline{[s=t]}$ shares no variables with $B \vee \underline{q[r]}$. The major premise of (6) is $B \vee q[r]$; $A \vee \underline{[s=t]}$ is the minor premise.

Remark. Notice that the kernel $\{[s=t], q[r]\}$ of (6) (§1.3.7) is defined by the designated literals in the premises, and that the residual $q[t]\theta$ is also the designated literal of the conclusion in (6). Thus, Rp is a restriction of paramodulation [62].

A Cut inference has the form

$$\{A \vee \underline{p},\ B \vee \underline{\tilde{q}}\} \vdash (A - p)\theta \vee (B - \tilde{q})\theta \qquad (7)$$

where θ is a m.g.u. of $\{p,q\}$, (7) is general provided that $A \vee \underline{p}$ shares no variables with $B \vee \underline{\tilde{q}}$.

Remark. Again, note the requirement that the kernel $\{p,\tilde{q}\}$ of (7) be the set of designated literals of premises. The residual of (7) is 0 .

2.2.3 Normal Embedding Transformations

Each basic inference is evidently a generalized resolution inference. Thus, the analysis of resolution-based deductions in §1.3.7 applies to the class of basic deductions. The following extension of the embedding transformation for resolution-based deductions will be quite useful in subsequent developments.

Let $\underline{\underline{\mathscr{D}}}'$ be a basic unit deduction from $\{p_i: i<m\}$. A normal embedding of $\underline{\underline{\mathscr{D}}}'$ in $\{B_k \vee \underline{q}_k: k<n\}$ is an embedding of $\underline{\underline{\mathscr{D}}}'$ in $\{A_i \vee \underline{p}_i: i<m\}$ (§1.3.7) where $A_i \vee \underline{p}_i$ is a simple factor of some clause $B_k \vee \underline{q}_k$ on q_k $(i=0,\cdots,n-1)$, prefixed by all SF-inferences $(\{B_k \vee q_k\} \vdash A_i \vee \underline{p}_i)$ used in obtaining $\{A_i \vee \underline{p}_i: i<m\}$ from $\{B_k \vee q_k: k<n\}$.

Thus, if $\underline{\underline{\mathscr{D}}}$ is a normal embedding of $\underline{\underline{\mathscr{D}}}'$ in $\{B_k \vee \underline{q}_k: k<n\}$ then $\mathrm{Base}(\underline{\underline{\mathscr{D}}}) \subset \{B_i \vee q_i: i<n\} \cup \{[x=x]\}^{\sim}$. Note that the designated literal of $B_i \vee q_i$ is irrelevant.

Remark. A normal deduction from $\{B_i \vee q_i: i<n\}$ is a Γ_S-deduction from $\{B_i \vee q_i: i<n\}$ and may therefore contain any number of variants of $[x=x]$ in its base (§1.3.3).

2.2.4 Analysis of Normal Deductions

A normal deduction is a basic deduction $\underline{\underline{\mathscr{D}}}$ wherein

(i) the premise of each SF-inference is either an initial (base) clause or the conclusion of a Cut inference ; and

(ii) each Rp or Cut inference is general.

Proposition 5. For each normal deduction $\underline{\underline{\mathscr{D}}}'$ there exists a general normal deduction $\underline{\underline{\mathscr{D}}}$ such that $\mathrm{Base}(\underline{\underline{\mathscr{D}}}) \leq \mathrm{Base}(\underline{\underline{\mathscr{D}}}')^{\sim}$.

Indication of Proof. Let π be a branch mapping (§1.3.8) for $(\underline{\underline{\mathcal{D}}}', \mathcal{A})$ where $\mathcal{A} \leq \text{Base}(\underline{\underline{\mathcal{D}}}')$. While Γ_S is not liftable, we nevertheless obtain a Γ_S-lifting $\underline{\underline{\mathcal{D}}}$ of $\underline{\underline{\mathcal{D}}}'$ based on $(\pi, \mathcal{A})$, for the reason that each basic inference in $\underline{\underline{\mathcal{D}}}'$ is general and therefore "lifts" to an inference whose induced substitution is a variant of the induced substitution of the inference being lifted. There is, in fact, a mapping from induced substitutions in inferences of $\underline{\underline{\mathcal{D}}}$ onto variant induced substitutions at corresponding positions in $\underline{\underline{\mathcal{D}}}'$. That $\underline{\underline{\mathcal{D}}}$ is general and $\text{Base}(\underline{\underline{\mathcal{D}}}) \leq \text{Base}(\underline{\underline{\mathcal{D}}}')^{\sim}$ follows by Proposition 7 in §1.3.8. Clearly $\underline{\underline{\mathcal{D}}}$ is normal.

A normal realization of a resolution inference

$$\{B_i \vee q_i : i < n\} \vdash C \tag{8}$$

where $(B_0 \vee \cdots \vee B_{n-1})\theta \geq C \geq (B_0\theta \vee q_0\theta) \vee \cdots \vee (B_{n-1}\theta \vee q_{n-1}\theta)$ is a normal embedding $\underline{\underline{\mathcal{D}}}(C)$ of a unit refutation $\underline{\underline{\mathcal{D}}}'(0)$ in $\{B_i \vee q_i : i < n\}$. Thus, $\text{Base}(\underline{\underline{\mathcal{D}}}'(0)) = \{p_i : i < m\}$ where $p_i = q_j\theta_j$ for some $j < n$ and some m.g.u. θ_j of a set of literals in $B_j \vee q_j$ which contains q_j.

Proposition 6. If $\underline{\underline{\mathcal{D}}}(C)$ is a normal realization of (8) then $\underline{\underline{\mathcal{D}}}(C)$ is normal and is a Γ_S-realization for (8).

The proof is trivial.

Proposition 7. Each normal refutation $\underline{\underline{\mathcal{D}}}(0)$ has a unique decomposition $\{\underline{\underline{\mathcal{D}}}_i(C_i) : i < n\}$ into normal realizations of resolution inferences.

Indication of proof. Use induction on the number n of Cut inferences in $\underline{\underline{\mathcal{D}}}(0)$.

Remark. Propositions 6 and 7 justify the analysis of normal deductions in terms of still simpler unit deductions, such as the class of $\mathcal{E}$-derivations which follows.

A derivation is a basic unit deduction consisting entirely of Rp-inferences.

An $\mathcal{E}$-derivation is a basic unit deduction

$$\begin{array}{ll} [s_0=t_0] & q_0[r_0] \\ & \vdots \\ [s_{n-1}=t_{n-1}] & q_{n-1}[r_{n-1}] \\ & q_n[r_n] \end{array} \qquad (9)$$

where $[s_i=t_i] \in \mathcal{E}^{\sim}$, $q_{i+1}[r_{i+1}] = q_i[t_i\theta_i]$, and θ_i is a simplest substitution such that $r_i = s_i\theta_i$ $(i=0,\cdots,n-1)$.

Remarks.

1. $(\{[s_i=t_i], q_i[r_i]\} \vdash q_i[t_i\theta_i])$ is an Rp-inference because θ_i is a m.g.u. of $\{r_i, s_i\}$ such that $q_i[t_i]\theta_i = q_i[t_i\theta_i]$.

2. For each $\mathcal{E}$-derivation (9) there is a corresponding general $\mathcal{E}$-derivation

$$\begin{array}{ll} [s_0=t_0]\eta_0 & q_0[r_0] \\ & \vdots \\ [s_{n-1}=t_{n-1}]\eta_{n-1} & \\ & q_n[r_n] \end{array} \qquad (10)$$

where η_j is an invertible substitution such that $[s_j=t_j]\eta_j$ shares no variables with $q_0[r_0]$ or with $\{[s_i=t_i]\eta_i: i<j\}$.

Recall the definition of $\sigma_{\underline{\underline{\mathcal{D}}}}$ in §1.3.7.

Proposition 8 (Induced Substitutions). Suppose $\underline{\underline{\mathcal{D}}}$ is a general basic deduction. Then $\sigma_{\underline{\underline{\mathcal{D}}}}$ is a m.g.s.u. of the set

$\{\{p,q\}$: $\underline{\underline{\mathcal{D}}}$ contains a factoring inference $B \vdash B\theta$, $\{p,q\} \subseteq B$, and $p\theta = q\theta\}$

$\cup\{\{p,q\}$: $\underline{\underline{\mathcal{D}}}$ contains a Cut-inference

$\{A \vee \underline{p}, B \vee \underline{\tilde{q}}\} \vdash (A\theta \vee B\theta)\}$

$\cup\{\{r,s\}$: $\underline{\underline{\mathcal{D}}}$ contains an Rp-inference $\{A \vee \underline{[s=t]}, B \vee \underline{q[r]}\} \vdash C \vee \underline{q[t]}\theta\}$.

Moreover, $\underline{\underline{\mathcal{D}}}\sigma_{\underline{\underline{\mathcal{D}}}}$ is a ground basic deduction with the same conclusion(s) as $\underline{\underline{\mathcal{D}}}$, and $\sigma_{\underline{\underline{\mathcal{D}}}}$ is a simplest substitution σ such that $\underline{\underline{\mathcal{D}}}\sigma$ has this property.

The derivation of this lemma from the results of §2.1.1 (by induction on the number of inferences in $\underline{\underline{\mathcal{D}}}$) is straightforward.

2.3 Resolution Micro-Refinements

2.3.1 Positional Order of Replacements in Derivations

Given a derivation from p to q based on a set $\mathcal{E}$ of equations, there are typically many alternative derivations from p to q based on $\mathcal{E}$. A good refinement should filter out all but one of these, so that proof procedures based upon it will not waste resources by generating many or all equivalent derivations "in parallel" while searching for the

first refutation.

Example. Let $\mathcal{E} = \{[a_i=b_i]: 1 \leq i \leq n\}$, and let $p = (Qa_1\cdots a_n)$ The "prefix" derivation

$$\begin{array}{ll} [a_1=b_1] & (Qa_1\cdots a_n) \\ & \quad\searrow \\ & (Qb_1a_2\cdots a_n) \\ & \quad\vdots \\ [a_n=b_n] & (Qb_1\cdots b_{n-1}a_n) \\ & \quad\searrow \\ & (Qb_1\cdots b_{n-1}b_n) \end{array}$$

is just one of $n!$ equivalent derivations from p to $(Qb_1\cdots b_n)$ based on $\mathcal{E}$. Even if subsumption were used, a proof procedure using a "complexity-order" search strategy would derive all of the 2^n atoms $(Qu_1\cdots u_n)$ where $u_i \in \{a_i,b_i\}$ $(i=1,\cdots,n)$ in the process of generating the first derivation from p to $(Qb_1\cdots b_n)$, unless some refinement were used to impose an ordering constraint on positions of occurrences replaced in "independent" Rp inferences (of which there are n above).

Such an ordering constraint is used in the definition of normal derivations and reductions (§2.3.5). The following total ordering on positions of occurrences in terms and literals is useful in this definition. (It formalizes the "left-to-right, bottom-to-top" order of positions in a familiar prefix-tree representation of terms).

End-order of positions (and occurrences at those positions) is defined (following Knuth [38]) by

(i) $\alpha\cdot\beta$ precedes α in end-order if $\beta \neq *$. (Thus, * is preceded by every other position, in end-order.)

(ii) $\alpha\cdot i\cdot\beta$ precedes $\alpha\cdot j\cdot\gamma$ in end-order provided $i < j$.

We say that an occurrence (u,α) precedes an occurrence (v,β) (in end-order) provided that α precedes β.

A position α dominates β provided that $\alpha \cdot \gamma = \beta$ for some position γ ; if $\gamma \neq *$ then α properly dominates β .

Two positions are independent provided that neither dominates the other.

It is sometimes useful to extend the relation of domination from positions to occurrences :

$$(u,\alpha) \text{ dominates } (v,\beta) =_{df} \alpha \text{ dominates } \beta .$$

The relation of independence is extended similarly.

2.3.2 Invariant Relations on Terms and Clauses

Subsequent developments make use of various ordering relations on terms and occasionally on clauses. Let R be a binary relation on $\mathfrak{T}_V$. The following attributes and transformations of R will be particularly useful.

Reflexive: $(t\,R\,t)$

Anti-reflexive: $(t\,\not R\,t)$

Transitive: If rRs and sRt then rRt

Symmetric: If sRt then tRs.

Anti-symmetric: If sRt and tRs then $s=t$.

Monotone: If sRt then $u[s]\,R\,u[t]$.

Substitutive: If sRt then $s\theta\,R\,t\theta$ $(\theta \in \Sigma_V)$.

Invariant: Monotone and substitutive.

Quasi-Ordering: Reflexive and transitive.

Partial-Ordering: Anti-symmetric and transitive[1]

---

[1]This definition of "partial ordering" leaves open the question of whether or not R is reflexive. (See Notational conventions below).

Total ordering: Partial ordering wherein either (sRt), (tRs), or (s=t) ($s,t \in \mathcal{T}_V$).

Well ordering: Total ordering wherein each subset of $\mathcal{T}_V$ has a first element.

A descending chain in R is a set $\{t_i : i \in N\}$ such that $t_i R t_{i+1}$ or $t_i = t_{i+1}$ $(i \in N)$.

Descending Chain Condition (D.C.C.): Every descending chain in R is finite.

Equivalence, congruence, and equality relations (§1.2.8) are easily defined in terms of the above concepts.

Notational conventions. Quasi-orderings will normally be denoted by $\Rightarrow$ or $\rightarrow^*$; $\rightarrow^*$ denotes the smallest reflexive and transitive extension of a generating relation $\rightarrow$. Partial orderings will normally be denoted by $\succ$ or $\succeq$, and it is normally assumed that $t \not\succ t$. $\succeq$ denotes the smallest reflexive extension of $\succ$, and is also a partial ordering. $\prec$ generally denotes either the converse of $\succ$ or a well-ordering unrelated to $\succ$.

Standard extensions to clauses. Frequently it is useful to tacitly extend a relation from $\mathcal{T}_V$ to $\mathcal{T}_V \cup \mathcal{C}_V$ without distinguishing the extension notationally. To this end we define two standard extensions of a relation from $\mathcal{T}_V$ to $\mathcal{T}_V \cup \mathcal{C}_V$. The monotone extension applies to all quasi-orderings. It has the property that if $\Rightarrow$ preserves an equality relation $=_\mathcal{E}$ on $\mathcal{T}_V$ then $\Rightarrow$ preserves $\models_\mathcal{E}$ on $\mathcal{C}_V$. The lexical extension is based on a standare well-ordering of V (precedes) where $=$ is the first member of $\{=\} \cup V_R$.

The monotone extension of a quasi-ordering $\Rightarrow$ on $\mathcal{T}_V$ is the smallest extension of $\Rightarrow$ from $\mathcal{T}_V$ to a quasi-ordering ($\Longrightarrow$) on $\mathcal{T}_V \cup \mathcal{C}_V$ satisfying (i)-(v):

(i) If $s \Rightarrow t$ then $u[s] \Rightarrow u[t]$

(ii) If $s_1 \Rightarrow t_1$ and $s_2 \Rightarrow t_2$ then $[s_1=t_1] \Rightarrow [s_2=t_2]$.

(iii) If $s_i \Rightarrow t_i$ $(i=1,\cdots,n)$ then $(Ps_1\cdots s_n) \Rightarrow (Pt_1\cdots t_n)$ $(P \in V_R^n)$.

(iv) If $p \Rightarrow q$ then $\tilde{p} \Rightarrow \tilde{q}$.

(v) If $p_i \Rightarrow q_i$ $(i=1,\cdots,n)$ then $(p_1 \vee \cdots \vee p_n) \Rightarrow (q_1 \vee \cdots \vee q_n)$.

The lexical extension of an anti-symmetric relation $\succ$ (normally not a quasi-ordering) on $\mathcal{T}_V$ is the smallest extension of $\succ$ from $\mathcal{T}_V$ to a relation ($\succ$) on $\mathcal{T}_V \cup \mathcal{C}_V$ which satisfies (a)-(d):

(a) $(Qu_1\cdots u_m) \succ (Pv_1\cdots v_n)$ iff either

(i) P precedes Q in the well-ordering of $V_R \cup \{=\}$; or

(ii) $Q = P$ and $u_j \succ v_j$ where $j = \min\{k: u_k \neq v_k\}$;

(b) If $q \succ p$ then $\tilde{q} \succ \tilde{p}$.

(c) If $q_i \succeq p_i$ $(i=1,\cdots,n)$ then $(q_1 \vee \cdots \vee q_n) \succeq (p_1 \vee \cdots \vee p_n)$.

(d) If $m < n$ then $(q_1 \vee \cdots \vee q_n) \succ (p_1 \vee \cdots \vee p_m)$.

It is easily verified that if $\succ$ is a partial ordering on $\mathcal{T}_V$ then its lexical extension is a partial ordering on $\mathcal{T}_V \cup \mathcal{C}_V$.

2.3.3 Reducibility Relations and Determinative Systems

An $\mathcal{E}$-reducibility relation is a quasi-ordering $\Rightarrow$ such that $(u \Rightarrow v)$ implies $(u =_{\mathcal{E}} v)$ $(u,v \in \mathcal{T}_V)$; $\Rightarrow$ is tacitly extended to $\mathcal{C}_V$

by the standard monotone extension transformation (§ 2.3.2), whence $(A \Rightarrow B)$ implies $(A \models_{\mathcal{E}} B)$.

Determinative systems[1]. A clause-set $\mathcal{E}$ is determinative provided that for each clause $A \vee [s=t]$ in $\mathcal{E}$, each variable which occurs in $A \vee [s=t]$ also occurs in s. A clause B is determinative provided that $\{B\}$ is determinative. Thus, an equation $[s=t]$ is determinative provided that each variable in t is also in s.

Determinative equations can naturally be interpreted as definitions. Two concepts of "definitional reduction" are formalized by the $\mathcal{E}$-reducibility relations $\rightarrow_{\mathcal{E}}^*$ and $\Rightarrow_{\mathcal{E}}^*$, respectively, generated by the relation $\rightarrow_{\mathcal{E}}$ of $\mathcal{E}$-contraction and its restriction $\Rightarrow_{\mathcal{E}}$, called normal $\mathcal{E}$-contraction:

$(u \rightarrow_{\mathcal{E}} v) =_{df}$ ($u = u'[r]$ and $v = u'[t\theta]$ where $\mathcal{E}$ contains a determinative equation $[s=t]$ such that $r = s\theta$).

$(u \Rightarrow_{\mathcal{E}} v) =_{df}$ ($u=u'[r]$ and $v=u'[t\theta]$ where $[r]$ is the first occurrence of a term r such that $\mathcal{E}$ contains a determinative equation $[s=t]$ and $r = s\theta$).

Observe that $\rightarrow_{\mathcal{E}}^*$ is an invariant $\mathcal{E}$-reducibility relation, whereas $\Rightarrow_{\mathcal{E}}^*$ is normally neither monotone nor substitutive.

One useful property of determinative systems is revealed by the following characterization of $=_{\mathcal{E}}$.

$\mathcal{E}$ is equational provided that each clause of $\mathcal{E}$ is an equation.

---

[1] Cf. the notion of determinative rule in Curry [15].

Proposition 9. Suppose $\mathcal{E}$ is determinative and equational. Then $u =_{\mathcal{E}} v$ iff there exists a list $(u_0,\cdots,u_n)$ such that $u_0 = u$, $u_n = v$, and either $u_k \to_{\mathcal{E}} u_{k+1}$ or $u_{k+1} \to_{\mathcal{E}} u_k$ $(k=1,\cdots,n)$.

Proof. Let $\sim_{\mathcal{E}}$ be the relation whereby $u \sim_{\mathcal{E}} v$ iff there exists such a list $(u_0,\cdots,u_n)$. Clearly $(u \sim_{\mathcal{E}} v)$ implies $(u =_{\mathcal{E}} v)$. Conversely, $\sim_{\mathcal{E}}$ is an invariant equivalence relation such that $s \sim_{\mathcal{E}} t$ for each equation $[s=t]$ in $\mathcal{E}$; since $=_{\mathcal{E}}$ is the smallest such relation, $(u =_{\mathcal{E}} v)$ implies $(u \sim_{\mathcal{E}} v)$.

Remark. If $\mathcal{E}$ is not determinative then $\sim_{\mathcal{E}}$ need not be invariant. For suppose $\mathcal{E} = \{[s=t]\}$ where y occurs in t but not in s . Then $s \sim_{\mathcal{E}} t$, but $s[u/y] \not\sim t[u/y]$ for $u \neq y$. Thus $\sim_{\mathcal{E}}$ is monotone but not substitutive.

2.3.4 Reduction Systems and Ordered Normal Forms

A reduction system is a pair $(\mathcal{E},\succ)$ where $\mathcal{E}$ is a (normally finite) set of clauses and $\succ$ is a monotone quasi-ordering on $\mathcal{T}_V$.

Conventions.

1. As in §2.1.3, $\underline{\underline{\mathcal{E}}}$ may denote either a reduction system $(\mathcal{E},\succ)$ or a triple $(\mathcal{E},\succ,s)$ where $(\mathcal{E},\succ)$ is a reduction system. Normally we assume $\underline{\underline{\mathcal{E}}} = (\mathcal{E},\succ)$.

2. If $\succ$ is an $\mathcal{E}$-reducibility relation (e.g., $\to^*_{\mathcal{E}}$) then we assume $\succ$ to have been extended to $\mathcal{T}_V$ by the standard monotone extension; otherwise we assume the (stronger) standard lexical extension (§2.3.2).

An $\underline{\underline{\mathcal{E}}}$-reducibility relation is an $\mathcal{E}$-reducibility relation $\Rightarrow^*$ such that $(u \Rightarrow^* v)$ implies $(u \succeq v)$. If $u \Rightarrow^* v$ where $v \neq u$ then

u is <u>reducible</u> in $\Rightarrow^*$: otherwise u is <u>irreducible</u> in $\Rightarrow^*$.

The following development parallels and extends §2.3.3, beginning with an <u>$\mathcal{E}$-contraction</u> relation ($\rightarrow_{\mathcal{E}}$) and a more restrictive <u>normal $\mathcal{E}$-contraction</u> relation ($\Rightarrow_{\mathcal{E}}$):

$(u \rightarrow_{\mathcal{E}} v) =_{df}$ ($u=u'[r]$ and $v = u'[t\theta]$ where $\mathcal{E}$ contains a determinative equation $[s=t]$ such that <u>$r = s\theta \succ t\theta$</u>)

$(u \Rightarrow_{\mathcal{E}} v) =_{df}$ ($u=u'[r]$ and $v = u'[t\theta]$ where $[r]$ is the <u>first</u> occurrence of a term r such that $\mathcal{E}$ contains a determinative equation $[s=t]$ and <u>$r = s\theta \succ t\theta$</u>).

Observe that u is irreducible in $\Rightarrow_{\mathcal{E}}^*$ iff u is irreducible in $\rightarrow_{\mathcal{E}}^*$.

$$NF(\mathcal{E},\succ) =_{df} \{u \in \mathcal{T}_V \cup \mathcal{C}_V : u \text{ is irreducible in } \rightarrow_{\mathcal{E}}^*\}$$

An <u>$\mathcal{E}$-normal form</u> (for $\mathcal{T}_V \cup \mathcal{C}_V$) is a decidable set $\mathcal{N} \subseteq \mathcal{T}_V \cup \mathcal{C}_V$ satisfying (i) and (ii):

(i) For each term s , $s \rightarrow_{\mathcal{E}}^* t$ for some term t in $\mathcal{N}$.

(ii) For each clause A, $A \rightarrow_{\mathcal{E}}^* B$ for some clause B in $\mathcal{N}$.

If t and B are uniquely determined by s and A in (i) and (ii) respectively, then $\mathcal{N}$ is an <u>$\mathcal{E}$-canonical form</u>.

Thus, $NF(\mathcal{E},\succ)$ may or may not be an <u>$\mathcal{E}$-normal form</u>. The following development is useful in investigations of such matters. We define pertinent attributes of a reduction system $\mathcal{E} = (\mathcal{E},\succ)$ as follows:

<u>Equational</u>: $\mathcal{E}$ is a set of equations.

<u>Determinative</u>: $\mathcal{E}$ is determinative.

<u>Finite</u>: $\mathcal{E}$ is finite.

Finitary: Finite, and $\succ$ is a partial ordering which satisfies D.C.C.

Total: $s \succ t$ for each equation $[s=t]$ in $\mathcal{E}$.

Complete: If $r =_{\mathcal{E}} s$ then $r \rightarrow^*_{\mathcal{E}} t$ and $s \rightarrow^*_{\mathcal{E}} t$ for some term t.

Normally Complete: If $r =_{\mathcal{E}} s$ then $r \Rightarrow^*_{\mathcal{E}} t$ and $s \Rightarrow^*_{\mathcal{E}} t$ for some term t.

Normal: $NF(\mathcal{E},\succ)$ is an $\mathcal{E}$-normal form.

Canonical: $NF(\mathcal{E},\succ)$ is an $\mathcal{E}$-canonical form.

Observe that if $\mathcal{E}$ is finitary then $\mathcal{E}$ is normal, and that $\mathcal{E}$ can be (normally) complete without being normal.

Proposition 10. Suppose $\mathcal{E}$ is finitary. Then $\mathcal{E}$ is complete iff $\mathcal{E}$ is canonical.

Proof. If $\mathcal{E}$ is canonical then clearly $\mathcal{E}$ is complete.

Suppose $\mathcal{E}$ is complete. It suffices to show that for each term s there exists a unique term t in $NF(\mathcal{E},\succ)$ such that $s \Rightarrow^*_{\mathcal{E}} t$. Suppose $s \Rightarrow^*_{\mathcal{E}} t_1$ and $s \Rightarrow^*_{\mathcal{E}} t_2$ where $t_1, t_2 \in NF(\mathcal{E},\succ)$. Then $t_1 =_{\mathcal{E}} t_2$, whence $t_i \Rightarrow^*_{\mathcal{E}} t$ $(i=1,2)$ for some t (by completeness). Therefore $t_1 = t = t_2$ by definition of $NF(\mathcal{E},\succ)$. ∎

This development is continued in §C.3 with an effective partial characterization of canonical reduction systems. (Completeness is not an effectively decidable attribute of reduction systems.)

Notice that $(\rightarrow_{\mathcal{E}}) = (\rightarrow_{\underline{\mathcal{E}}})$ and $(\Rightarrow_{\mathcal{E}}) = (\Rightarrow_{\underline{\mathcal{E}}})$ where $\underline{\mathcal{E}} = (\mathcal{E}, \rightarrow^*_{\mathcal{E}})$. In this sense, $\rightarrow_{\mathcal{E}}$ and $\Rightarrow_{\mathcal{E}}$ are special cases of the above concepts defined for reduction systems.

Convention. Whenever a concept is defined with a reduction-system parameter $\underline{\mathcal{E}}$, substitution of $\mathcal{E}$ in the definition is equivalent to substitution of the reduction-system parameter $(\mathcal{E}, \rightarrow_{\mathcal{E}}^*)$.

Thus, the concepts total, normal, canonical, finitary, and complete apply to sets of clauses as well as to reduction systems; moreover, $NF(\mathcal{E})$ is defined:

$$NF(\mathcal{E}) =_{df} NF(\mathcal{E}, \rightarrow_{\mathcal{E}}^*)$$

Similarly, following the useful definition

$$\Sigma_{\underline{\mathcal{E}}} =_{df} \{\theta \in \Sigma_V : x\theta \in NF(\mathcal{E}, \succ) \quad (x \in V_I)\}$$

the substitution set $\Sigma_{\mathcal{E}}$ is also defined.

Observe that $\mathcal{E}$ is total because $\rightarrow_{\mathcal{E}}^*$ orders each equation of $\mathcal{E}$.

2.3.5 Complexity Orderings and Weighting Functions

An $\mathcal{E}$-complexity ordering is a decidable monotone partial ordering $\succ$ on $\mathcal{T}_V$ satisfying (i)-(iv):

(i) $\succ$ well-orders the constant terms of $\mathcal{T}_V$.

(ii) $u[t] \succeq t$

(iii) If $s \succ t$ then $s\theta \succ t\theta$ $(\theta \in \Sigma_{\mathcal{E}})$

(iv) $NF(\mathcal{E})$ contains at least one constant term.

Notation. If $\mathcal{E} = 0$ then the prefix $\mathcal{E}$- is omitted.

Remark. If $\succ$ is a complexity ordering then $\Sigma_0 = \Sigma_V$ and $\succ$ is invariant, whence $\succ$ satisfies D.C.C. by (i) and (iii). More generally, if $s \succ t$ then $s\theta \succ t\theta$ where $x\theta$ is a constant term in $NF(\mathcal{E})$ (using (iv)). It follows that every $\mathcal{E}$-complexity ordering

satisfies D.C.C. Thus, by Proposition 9 we have

Proposition 11. If $\mathcal{E}$ is finite and determinative, and $\succ$ is an $\mathcal{E}'$-complexity ordering, then $NF(\mathcal{E},\succ)$ is an $\mathcal{E}$-normal form.

Remark. Subsequent completeness results for $ND(\mathcal{E},\succ)$ presuppose that $\succ$ is an $\mathcal{E}'$-complexity ordering where $\mathcal{E}' \leq \mathcal{E}$.

A weighting function for V is a mapping $w: V_F \to N$ such that

(i) $w(c) > 0$ $(c \in V_F^0)$; and

(ii) $w(f) > 0$ for all f in V_F^1 except for the last member of V_F , if any, in the standard well-ordering of V .

Weighting functions can be used to define complexity orderings for $\mathcal{T}_V$ in several ways. The basic method follows. Applications and extensions of this method are investigated in §C and §D .

Let w be a weighting function for V and let $w_0 =_{df} \min\{w(c): c \in V_F^0\}$. Extend w to $\mathcal{T}_V$ by

$$w(u) =_{df} \begin{cases} w_0 , & \text{if } u \in V_I ; \\ w(f) + \sum_{i=1}^{n} w(u_i), & \text{if } u = (fu_1 \cdots u_n) \text{ where } f \in V_F^n \ (n \geq 0). \end{cases}$$

Let $n(x,u)$ be the number of occurrences of x in u , and let the first symbol of u be u provided that $u \in V_I \cup V_F^0$ and f provided that $u = (fu_1 \cdots u_n)$ where $f \in V_F^n$ and $n > 0$:

$$op(u) =_{df} \text{the first symbol of } u.$$

Now $\succ_w$ is defined from w and the standard well-ordering of V (precedes) wherein operation constants precede variables:

$(u >_W v) =_{df}$ [If $n(x,u) \geq n(x,v)$ $(x \in V_I)$

then if $w(u) > w(v)$

then true

else if $w(u) = w(v)$

then if $v \in V_I$

then $(u \neq v)$

else if op(v) precedes op(u)

then true

else if $u = fu_1 \cdots u_n$

$\neq v = fv_1 \cdots v_n$

and $u_j >_W v_j$

where $j = \min\{k: u_k \neq v_k\}$

then true

else false

else false

else false]

It is easily verified that $>_W$ is a complexity ordering on $\mathcal{T}_V$.

(See Knuth and Bendix [39] for proof.)

2.3.6 $\mathcal{E}$-normal Reductions

Let $\mathcal{E}$ be a reduction system (§2.3.4).

An $\mathcal{E}$-normal reduction is an $\mathcal{E}$-derivation

$$\begin{array}{ll} [s_0=t_0] & q_0[r_0] \\ & \vdots \\ [s_{n-1}=t_{n-1}] & q_{n-1}[r_{n-1}] \\ & q_n[r_n] \end{array} \qquad (10)$$

where

(i) $t \in NF(\mathcal{E},\succ)$ for each term t which occurs in $q_i[r_i]$ before $[r_i]$ (end-order);

(ii) $r_i = s_i\theta_i \succ t_i\theta_i$; and

(iii) if $r_i = r_j$ then $[s_i=t_i]$ is a variant of $[s_j=t_j]$.

(10) is <u>complete</u> provided that, in addition,

(iv) $q_n[r_n] \in NF(\mathcal{E},\succ)$.

<u>Remarks</u>

1. If $\succ$ satisfies D.C.C. then every $\underline{\mathcal{E}}$-normal reduction has an extension to a complete $\underline{\mathcal{E}}$-normal reduction, because $q_i[r_i] \succ q_{i+1}[r_{i+1}]$ by monotonicity of $\succ$.

2. If (10) is an $\underline{\mathcal{E}}$-normal reduction then $q_0[r_0] \Rightarrow^*_{\underline{\mathcal{E}}} q_n[r_n]$.

2.3.7 <u>$\mathcal{E}$-normal Inferences</u>

$\underline{\mathcal{E}}$-normal inferences are effective inferences defined in terms of a reduction system $\underline{\mathcal{E}} = (\mathcal{E},\succ)$. Their definition facilitates the definition of $\underline{\mathcal{E}}$-normal deductions in §2.3.8. Two sorts of $\underline{\mathcal{E}}$-normal inferences are described below:

(i) <u>Basic $\mathcal{E}$-normal inferences</u> consist of SF, $Rp_{\underline{\mathcal{E}}}$, and $Cut_{\underline{\mathcal{E}}}$.

(ii) <u>Composite $\mathcal{E}$-normal inferences</u> are realized by basic $\mathcal{E}$-normal inferences, and are the inferences most naturally produced by refutation procedures "based on" $ND(\mathcal{E},\succ)$ (§3.2.3).

$\underline{Rp_{\mathcal{E}}}$: $\{A \vee \underline{[s=t]},\ B \vee q[r]\} \vdash A\theta \vee B\theta \vee \underline{q[t]\theta}$ where

(i) $A \vee \underline{[s=t]}$ shares no variables with $B \vee \underline{q[r]}$;

(ii) θ is a m.g.u. of $\{r,s\}$;

(iii) $[s=t] \in NF(\mathcal{E} - [s=t]^{\sim}, \succ)$;

(iv) if $[s=t]$ is properly subsumed by an equation $[u=v]$ in $\mathcal{E}$ then $r = s\theta \succ t\theta$ and $q[r]\theta = q[s\theta]$,

(v) if $q[r] \notin NF(\mathcal{E}, \succ)$ then $r = s\theta \succ t\theta$, $q[r]\theta = q[s\theta]$, and each term which occurs in $q[r]$ before $[r]$ is in $NF(\mathcal{E}, \succ)$;

(vi) if $q[r] \in NF(\mathcal{E}, \succ) - [x=x]^*$ then $r \notin V_I$ and $t\theta \not\succ s\theta$; and

(vii) if $q[r] \in [x=x]^{\sim}$ then $B = 0$, $[s=t] \in NF(\mathcal{E}, \succ)$, and $r\theta = s \not\succ t$.

$\underline{\underline{Cut}}_{\mathcal{E}}$: $\{A \vee \underline{p}, B \vee \tilde{\underline{q}}\} \vdash A\theta \vee B\theta$

where

(i) $A \vee \underline{p}$ shares no variables with $B \vee \tilde{\underline{q}}$;

(ii) θ is a m.g.u. of $\{p,q\}$;

(iii) $p,q \in NF(\mathcal{E}, \succ)$; and

(iv) if either p or $\tilde{q}$ is an equation, then $\{p,\tilde{q}\} = \{[y=y], [u \neq v]\}$ and θ is a m.g.u. of $\{u,v\}$.

Composite $\mathcal{E}$-normal inferences are of the forms SF, $\underline{\underline{CRp}}_{\mathcal{E}}$, and $\underline{\underline{CCut}}_{\mathcal{E}}$:

$\underline{\underline{CRp}}_{\mathcal{E}}$: $\{A \vee p, B \vee q\} \vdash A\theta \vee B\theta \vee \underline{q'[t]\theta}$

where

(i) either $p = [s=t]$ and $q \Rightarrow^*_{\mathcal{E}} q'[t]$; or

$p \Rightarrow^*_{\mathcal{E}} [s=t]$ and $q = q'[r]$ or p; or

$p \Rightarrow^*_{\mathcal{E}} [t=s]$ and $q = q'[r]$; and

(ii) $(\{A \vee [s=t], B \vee \dot{q}[r]\} \vdash A\theta \vee B\theta \vee \underline{q'[t]\theta})$

is an $Rp_{\underline{\mathcal{E}}}$-inference.

$\underline{CCut}_{\underline{\mathcal{E}}}$: $\{A \vee p, B \vee \tilde{q}\} \vdash A\theta \vee B\theta$

where

(i) $p \Rightarrow^*_{\underline{\mathcal{E}}} p' \in NF(\mathcal{E},\succ)$; and

(ii) $(\{A \vee p', B \vee \tilde{q}\} \vdash A\theta \vee B\theta)$ is a $\underline{Cut}_{\underline{\mathcal{E}}}$-inference.

2.3.8 $\underline{\mathcal{E}}$-normal Deductions

An $\underline{\mathcal{E}}$-normal deduction is a normal deduction whose constituent normal resolution inference realizations (§2.2.3) are embeddings of unit $\underline{\mathcal{E}}$-normal deductions. Thus, the class $ND(\mathcal{E},\succ)$ of $\underline{\mathcal{E}}$-normal deductions is a resolution micro-definement defined in terms of unit $\underline{\mathcal{E}}$-normal deductions.

Define $S(p)$ by

$$S(p) =_{df} \begin{cases} p^* \text{ , if } p \text{ is not an equation;} \\ \{[u-v] \in [s=t]^*: \text{ either some variable of } v \text{ is not in } u \text{ or else } v = t\eta \text{ where } \eta \text{ is the simplest substitution such that } u=s\eta\} \text{ , if } p \text{ is an equation } [s=t]. \end{cases}$$

$S(p)$ is the set of literals q such that p subsumes q and p can be used in any normal inference where p can be used. Observe that if $p = [fx=gy]$ and $fx \succ gx$ then $[fx=gx] \notin S(p)$ because $[fx=gx]$ can be used in an $\underline{\mathcal{E}}$-normal reduction whereas $[fx=gy]$ cannot.

A (basic) derivation $\underline{\mathcal{D}}$ is $\underline{\mathcal{E}}$-normal provided that $\underline{\mathcal{D}}$ has a decomposition $\{\underline{\mathcal{D}}_i(q_i): i<n\}$ satisfying (i)-(iii):

(i) $\text{Base}(\underline{\mathcal{D}}_k(q_k)) \subseteq \text{Base}(\underline{\mathcal{D}}) \cup \{q_j: j<k\}$ $(k=0,\cdots,n-1)$.

(ii) If q_k is not a conclusion of $\underline{\mathcal{D}}$ then $\underline{\mathcal{D}}_k(q_k)$ realizes a unit $\text{CRp}_{\underline{\mathcal{E}}_k}$-inference $(\{p,q\} \vdash q_k)$ where $\mathcal{E}_k$ is defined by $\mathcal{E}_k =_{df} \mathcal{E} \cup \text{Base}(\underline{\mathcal{D}}) \cup \{q_j: j<k\}$ $(k=0,\cdots,n)$.

(iii) If $\underline{\mathcal{D}}_k(q_k)$ does not realize a unit $\text{CRp}_{\underline{\mathcal{E}}_k}$-inference then $\underline{\mathcal{D}}_k(q_k)$ is an $\underline{\mathcal{E}}_k$-normal reduction.

(iv) If $p,q \in \text{Base}(\underline{\omega})$ and p subsumes q then p and q are variants.

(v) If $q_k \in S(p)$ where $p \in \mathcal{E}_k$ then q_k is a conclusion of $\underline{\mathcal{D}}$.

(vi) If $p \in S(q_k)$ and $q_k \notin S(p)$ then $p \notin \text{Base}(\underline{\mathcal{D}}_j(q_j))$ $(j=k+1,\cdots,n-1)$.

A unit refutation $\underline{\mathcal{D}}$ is $\underline{\mathcal{E}}$-normal provided that $\underline{\mathcal{D}}$ has a decomposition $\{\underline{\mathcal{D}}_i(q_i): i<n\} \cup \{\underline{\mathcal{D}}_n(0)\}$ where

(vii) The subdeduction $\underline{\mathcal{D}}'$ composed of $\{\underline{\mathcal{D}}_i(q_i): i<n\}$ is an $\underline{\mathcal{E}}$-normal derivation; and

(viii) $\underline{\mathcal{D}}_n(0)$ is a $\text{CCut}_{\underline{\mathcal{E}}_n}$-inference realization.

An $\underline{\mathcal{E}}$-normal realization of a resolution inference (8) is a normal embedding $\underline{\mathcal{D}}(C)$ of an $\underline{\mathcal{E}}$-normal refutation $\underline{\mathcal{D}}'(0)$ in $\{B_i \vee \underline{q}_i: i<n\}$.

A normal deduction $\underline{\mathcal{D}}$ is $\underline{\mathcal{E}}$-normal provided that $\underline{\mathcal{D}}$ has a decomposition $\{\underline{\mathcal{D}}_i(C_i)\ i<n\}$ satisfying (i)-(vi):

(i) $\text{Base}(\underline{\mathcal{D}}_k(C_k) \subseteq \text{Base}(\underline{\mathcal{D}}) \cup \{C_j: j<k\}$ $(k=0,\cdots,n-1)$.

(ii) If C_k is not a conclusion of $\underline{\mathcal{D}}$ then $\underline{\mathcal{D}}_k(C_k)$ is an

$\underline{\mathcal{E}}_k$-normal realization of a resolution inference with premises in $\text{Base}(\underline{\mathcal{D}}) \cup \{C_j : j < k\}$, where $\underline{\mathcal{E}}_k =_{df} \mathcal{E} \cup \text{Base}(\underline{\mathcal{D}}) \cup \{C_j : j < k\}$.

(iii) If $\underline{\mathcal{D}}_k(C_k)$ is not an $\underline{\mathcal{E}}_k$-normal realization of a resolution inference then $\underline{\mathcal{D}}_k(C_k)$ is a normal embedding of an $\underline{\mathcal{E}}_k$-normal derivation in $\text{Base}(\underline{\mathcal{D}}_k(C_k))$; moreover, $\underline{\mathcal{D}}_j(C_j)$ is not an $\underline{\mathcal{E}}_j$-normal realization of a resolution inference $(j-k+1,\cdots,n-1)$.

(iv) If $A,B \in \text{Base}(\underline{\mathcal{D}})$ and A subsumes B then A and B are variants.

(v) If $\underline{\mathcal{D}}_k(C_k)$ is an $\underline{\mathcal{E}}_k$-normal realization of a resolution inference and C_k is subsumed by B in $\mathcal{E}_k$ then either C_k is a conclusion of $\underline{\mathcal{D}}$ or C_k and B are equations such that $C_k \notin S(B)$.

(vi) If $\underline{\mathcal{D}}_k(C_k)$ is an $\underline{\mathcal{E}}_k$-normal realization of a resolution inference and C_k subsumes B then either $B \notin \text{Base}(\underline{\mathcal{D}}_j(C_j))$ $(j=k+1,\cdots,n-1)$ or C_k and B are equations such that $B \notin S(C_k)$.

The class of normal $\underline{\mathcal{E}}$-deductions is defined by

$$ND(\mathcal{E},\succ) =_{df} \{\underline{\mathcal{D}} : \underline{\mathcal{D}} \text{ is } \underline{\mathcal{E}}\text{-normal}\}$$

Composite $\underline{\mathcal{E}}$-normal deductions are defined essentially as above (with SF restricted to initial clauses and conclusions of CCut - inferences), but on the basis of the composite $\mathcal{E}$-normal inference rules (§2.3.7) instead of the basic ones.

$$CND(\mathcal{E},\succ) =_{df} \{\underline{\mathcal{D}} : \underline{\mathcal{D}} \text{ is a composite } \underline{\mathcal{E}}\text{-normal deduction}\}.$$

Remarks.

1. $CND(\mathcal{E},\succ)$ is a non-basic $\mathcal{E}$-resolution micro-refinement; however, a member $CND(\mathcal{E},\succ)$ can be effectively "interpolated" so as to obtain a corresponding member of $ND(\mathcal{E},\succ)$.

2. Unit subsumption-deletion constraints are imposed within resolution-inference realizations in $ND(\mathcal{E},\succ)$ by (iv)-(vi) in the definition of $\underline{\mathcal{E}}$-normal derivations and unit refutations.

3. A similar set of subsumption-deletion constraints is imposed by (iv)-(vi) in the definition of $\underline{\mathcal{E}}$-normal deduction.

2.4 Normal Refinements and Their Proof Procedures

2.4.1 Normal Compositions

Recall that a generalized $\mathcal{E}$-resolution refinement is a class Δ of deductions composed of generalized $\mathcal{E}$-resolution inferences (§1.3.7):

$$\{B_i: i<n\} \vdash (B_0 - C_0)\theta \vee \cdots \vee (B_{n-1} \vee C_{n-1})\theta \vee C_n\theta \qquad (11)$$

where $C_i \subseteq B_i$ $(i=0,\ldots,n-1)$ and $\{C_i\theta: i<n\} \models_{\mathcal{E}} C_n\theta$. The kernel inference of (11) is the generalized $\mathcal{E}$-resolution inference

$$\{C_i: i<n\} \vdash C_n\theta \qquad (12)$$

Given a generalized $\mathcal{E}$-resolution refinement Δ and a generalized $\mathcal{E}'$-resolution refinement Δ' where $\mathcal{E} \supseteq \mathcal{E}'$, a normal composition of Δ and Δ' is a generalized $\mathcal{E}'$-resolution refinement $\Delta\cdot\Delta'$ wherein each deduction $\underline{\mathcal{D}}$ has a decomposition $\{\underline{\mathcal{D}}_i(C_i): i<n\}$ satisfying (i)-(iv):

(i) $\text{Base}(\underline{\underline{\mathcal{D}}}_j(C_j)) \subseteq \text{Base}(\underline{\underline{\mathcal{D}}}) \cup \{C_i\colon i<j\}$ $(j=0,\ldots,n-1)$.

(ii) If C_i is not a conclusion of $\underline{\underline{\mathcal{D}}}$ then $\underline{\underline{\mathcal{D}}}_i(C_i)$ realizes a generalized $\mathcal{E}$-resolution inference

$$\{B_k^i\colon k<n\} \vdash C_i \tag{13}$$

with kernel inference

$$\{C_k^i\colon k<n_i\} \vdash C_{n_i}^i \tag{14}$$

(iii) $\{(\{B_k^i\colon k<n_i\} \vdash C_i)\colon C_i$ is not a conclusion of $\underline{\underline{\mathcal{D}}}\}$ defines (the set of inferences in) a deduction in Δ .

(iv) If C_i is not a conclusion of $\underline{\underline{\mathcal{D}}}$ then (13) is realized by embedding in $\{B_k^i\colon k<n_i\} \cup \mathcal{E}^{\omega}$ a generalized $\mathcal{E}'$-resolution deduction $\underline{\underline{\mathcal{D}}}_i'(C_{n_i}^i)$ realizing (14), where $\underline{\underline{\mathcal{D}}}_i'(C_{n_i}^i)$ is in Δ' and $\text{Base}(\underline{\underline{\mathcal{D}}}_i'(C_{n_i}^i)) \subseteq \{C_k^i\colon k<n_i\} \cup \mathcal{E}^{\omega}$.

Remark. It is not appropriate to define a unique normal composition $\Delta\cdot\Delta'$ for the following reasons:

(a) If we take $\Delta\cdot\Delta'$ to be the set of all subdeductions of refutations in $\Delta\cdot\Delta$, then $\Delta\cdot\Delta'$ is not necessarily decidable in the class of all $\mathcal{E}'$-resolution inferences--i.e., $\Delta\cdot\Delta'$ is not necessarily a refinement.

(b) If we take $\Delta\cdot\Delta'$ to be the class of all $\mathcal{E}'$-resolution deductions $\underline{\underline{\mathcal{D}}}$ satisfying (vi)-(ix), then we may be admitting many deductions which, due to specific

information we have about Δ obviously cannot be extended to complete deductions in $(\Delta\cdot\Delta')^-$.

It is also unnecessary to define a unique normal composition at the present level of formal analysis.

For the purposes of this report it is necessary to understand normal composition in two basic cases:

Case 1: Δ is an $\mathcal{E}$-resolution refinement HR($\mathcal{E}$,$\succ$,s) and Δ' is an $\mathcal{E}'$-resolution refinement HR($\mathcal{E}'$,$\succ'$,s'). Then $C^i_k\theta$ is an r_s-positive literal for $k=0,\cdots,n_i-2$, $C^i_{n_i-1}$ is an r_s-negative literal, and $C^i_{n_i} = 0$. (In the similar case of hyper-$\underline{\mathcal{E}}$-resolution, $C^i_{n_i-1}$ is an r_s-negative clause.)

Case 2: Δ is an $\mathcal{E}$-resolution refinement where $\mathcal{E} \subseteq \mathcal{L}_V$, and Δ' is an $\mathcal{E}$-resolution micro-refinement, such as ND($\mathcal{E}$,$\succ$) or CND($\mathcal{E}$,$\succ$), wherein each deduction is normal in the sense that (simple) factoring applies only to initial clauses and conclusions of Cut inferences. Then again $C^i_k\theta$ is a literal and $C^i_{n_i} = 0$. In this case, $\underline{\mathcal{D}}^!_i(0)$ is simply a normal refutation of $\{C^i_k: k<n_i\} \cup \mathcal{E}$.

2.4.2 Normal Refinements

A normal refinement is a normal composition $\Delta_M\cdot\Delta_\mu$ where Δ_M is an $\mathcal{E}$-resolution refinement, Δ_μ is an $\mathcal{E}$-resolution micro-refinement, and $\mathcal{E}$ is a finite subset of $\mathcal{L}_V$.

Convention. If Δ_μ is a basic refinement then it is also a resolution micro-refinement, and Δ_M is easily formulated as a resolution refinement instead of an $\mathcal{E}$-resolution refinement. Thus, there is no loss of generality in supposing that Δ_M is a resolution refinement and Δ_μ is a basic resolution micro-refinement.

2.4.3 Normal Refutation Procedures

A normal refutation procedure is a refutation procedure Π for a pair (Γ,Δ) where Δ is a normal refinement $\Delta_M \cdot \Delta_\mu$ (§2.4.2). The structured programming of normal refutation procedures based on the schema Ref is outlined in §1.1.4. Thus, we assume below that $\Pi = \text{Ref}[E/\text{Enq}]$ where $E = E_M \cdot E_\mu$, the enqueuing function obtained by "composing" enqueuing functions E_M for Δ_M and E_μ for Δ_μ. Recall that Ref uses $\Delta = \Delta_M \cdot \Delta_\mu$ where $\Delta_M = (\Delta_n \cdot \cdots \cdot \Delta_0)$, Δ_k is an $\mathcal{E}_k$-resolution refinement, and $\mathcal{E} = \mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ where $\mathcal{E}_0 \subseteq \mathcal{E} \cap \mathcal{L}_V$.

The purpose of this subsection is to suggest computationally efficient representations for Δ and for E. Being presently concerned with the design of refinements, I shall make only a few plausible assumptions and remarks about E.

A representation for Δ. Recall that Ref uses Res_Δ, a procedural representation for Δ. The real problem before us is the design of a computationally efficient realization for Res_Δ on the basis of $\Delta_n,\cdots,\Delta_0$, and Δ_μ. The solution outlined below is based upon an analysis of the ancestors of each literal in a clause of R, the "current deduction" of Ref(C) (§2.2.3). This analysis of ancestors relies on the following definition:

$$\hat{\mathcal{E}}_k =_{df} \mathcal{E}_k - \{B: B \in \mathcal{B} \text{ for some } \mathcal{E}_k\text{-resolution inference } (\mathcal{B} \vdash C) \text{ occurring in a member of } \Delta_k\} .$$

Thus, if $\Delta_k = HR(\mathcal{E}_k, \succ_k, s_k)$ then $\hat{\mathcal{E}}_k = \{B \in \mathcal{E}_k: B \text{ is non-negative under the remaining } r_s\}$.

Note. In the following definition, the residual $q[t]\theta$ in an Rp-inference $\{A \vee [s=t], B \vee q[r]\} \vdash A \vee B \vee q[t]\theta$ is considered to be a descendant of $q[r]$; otherwise descendant is extended reflexively and transitively as in §1.3.7.

Levels of clauses. The following description of Res_Δ can be clarified by representing each clause C of a deduction $\mathcal{D}$ in Δ in the form

$$C = (C_n \vee \cdots \vee C_\ell) \tag{12}$$

where, relative to the proof-tree $\mathcal{D}(C)$,

(i) C_n is the set of descendants of literals[1] from initial clauses in $\mathcal{C}^\sim - \hat{\mathcal{E}}_n^*$ (§2.2.3);

(ii) C_k is the set of additional descendants of literals from initial clauses in $\mathcal{E}_{k+1}^\sim - \hat{\mathcal{E}}_k^*$ $(k=n-1,\cdots,\ell)$; and

(iii) $C_\ell \neq 0$.

The current level of C is ℓ :

$$Level_{\mathcal{D}}(C) =_{df} \ell \text{ where } C \text{ satisfies (12) (i-iii).}$$

---

[1] That is, C_n is the set of literals whose atoms are descendants of atoms occurring in ....

Notice that $j \neq k$ implies $C_j \cap C_k = 0$ due to (ii). Moreover the designated literal of C must be in C_ℓ, because if $\ell < n$ then any $\mathcal{E}_{\ell+1}$-resolution inference realization of which C is a constituent must be completed as a subdeduction of any $\mathcal{E}_k$-resolution inference realization of which C is a constituent ($k=\ell+2,\cdots,n$).

$\mathcal{E}_k$-resolution premises. If $\underline{\underline{\mathcal{D}}}$ is a refutation in Δ then $\underline{\underline{\mathcal{D}}}$ has a decomposition into realizations of $\mathcal{E}_k$-resolution inferences, the premises and designated literals of these inferences being determined by $\Delta_n \cdot \cdots \cdot \Delta_k$ ($k=0,\cdots,n$). Thus, for any clause C in $\mathcal{D}$, we may select the nearest set of ancestors of C in $\underline{\underline{\mathcal{D}}}$ which are in the set of premises of the $\mathcal{E}_k$-resolution inference realization which contains C. These are called the $\mathcal{E}_k$-resolution premises of C in $\underline{\underline{\mathcal{D}}}$.

Now suppose $\underline{\underline{\mathcal{D}}} \in \Delta$. Using the Level function, it is not difficult to define for each clause C in $\mathcal{D}$ a set $\mathcal{B}_k$ of ancestors of C which will be the $\mathcal{E}_k$-resolution premises of C in any extension of $\underline{\underline{\mathcal{D}}}$ to a refutation in Δ. $\mathcal{B}_k$ is the set of $\mathcal{E}_k$-resolution premises of C in $\underline{\underline{\mathcal{D}}}$. It turns out that $\text{Level}_{\underline{\underline{\mathcal{D}}}}(B) \geq k$ for each member of $\mathcal{B}_k$. Indeed, $\mathcal{B}_k$ is the set of nearest ancestors B of C in $\underline{\underline{\mathcal{D}}}$ such that $\text{Level}(B)_{\underline{\underline{\mathcal{D}}}} \geq k$ and B is (a simple factor of) a member of $\text{Base}(\underline{\underline{\mathcal{D}}})$ or a conclusion of a Cut-inference.

Given clauses A,B in $\mathcal{D}$ such that $(\{A,B\} \vdash C)$ is an admissible Rp- or Cut-inference based on $\underline{\underline{\mathcal{D}}}$ according to Δ_μ, we can use Δ_k ($\text{in}(\Delta_n \cdot \cdots \cdot \Delta_k)$) to determine whether the union of the $\mathcal{E}_k$-resolution premises of A and the $\mathcal{E}_k$-resolution premises of B in $\underline{\underline{\mathcal{D}}}$ could possibly be included in the premises of some $\mathcal{E}_k$-resolution

inference realized by some extension of $\underline{\mathscr{D}}$ in Δ (k=n,···,min{Level$_{\underline{\mathscr{D}}}$(A), Level$_{\underline{\mathscr{D}}}$(B)). If the answer is no for any of these choices of k , then the inference ({A,B} $\vdash$ C) should be rejected by Res_{Δ} .

Factors. Constraints on SF-inferences in Δ are equally easy to obtain from $\Delta_0,\cdots,\Delta_n$. Basically, the premise of each SF-inference {A v B} $\vdash$ Aθ v $\underline{B\theta}$ in a member $\underline{\mathscr{D}}$ of Δ must either be in Base($\underline{\mathscr{D}}$) or be the conclusion of a Cut-inference. The choice of key literal is determined by $\Delta_{\text{Level}_{\underline{R}}(A \vee B)}$, and $B \subseteq C_\ell$ where $A \vee B = (C_n \vee \cdots \vee C_\ell)$ satisfying (i)-(iii) following (12).

Composition of enqueuing functions. The basic fact to remember is that (ideally) E_n was designed by an "expert" so as to quickly lead to refutations of $\mathcal{E}$-inconsistent sets, and that E_k was designed by an "expert" at refuting $\mathcal{E}_k$-inconsistent sets $\{q_1,\cdots,q_m\} \cup (\tilde{\mathcal{E}}_{k+1} - \hat{\mathcal{E}}^*_k)$. Thus, once E_n has "decided" that a clause A of level n should be selected from Q as a premise for a "useful" $\mathcal{E}_n$-resolution inference, E_{n-1} should be used to decide what other clauses of Q , if any, should be brought in to efficiently complete a realization for the $\mathcal{E}_n$-resolution inference by $\mathcal{E}_{n-1}$-resolution inferences. E_{n-1} will in turn relinquish control to E_{n-2}, etc., until finally all of the lower level search strategies have been hierarchically invoked in "deciding" how best to realize a good $\mathcal{E}$-resolution inference based on the current deduction.

Enqueuing functions and merit orderings. For present purposes it is assumed that Enq is a procedure which defines an enqueuing function (§1.3.6). Thus Enq(Q,T) may use other parameters of Ref ,

such as $\underline{R}$ (the current deduction), $\mathcal{C}$ (the input premises) and $\mathcal{E}$ (the axioms), in computing the sequence $Enq(Q,T)$. Specifically, it is assumed that $Enq(Q,T)$ is ordered in decreasing order of merit, as defined by a completion cost estimation function $f: \mathcal{C}_V \rightarrow$ (Non-negative Reals $\cup \{\infty\}$). Intuitively the merit of B is inversely proportional to its completion cost (relative to Q, $\underline{R}$, and $\Delta(\mathcal{C} \cup \mathcal{E} \cup [x=x])$), which is the minimal cost of a complete "extension" $\underline{\underline{\mathcal{D}}}$ of $\underline{R}$ (in $\Delta^{-}(\mathcal{C} \cup \mathcal{E} \cup [x=x])$) which contains B. Thus the merit of O should be maximal, and the merit of a clause which cannot contribute to a refutation should be less than the merit of one which can.

Completion cost estimators. On the basis of previous research [58,74,40], it is reasonable to assume that f is defined on clauses of Q by

$$f(B) = g(\underline{\underline{R}}(B)) + h(Q,\underline{\underline{R}},B) \qquad (13)$$

where g defines the "cost" of the proof-tree for B and $h(Q,\underline{\underline{R}},B)$ estimates the (additional) minimal cost of a complete deduction $\underline{\underline{\mathcal{D}}}$ in $\Delta^{-}(\mathcal{C} \cup \mathcal{E} \cup [x=x])$, containing B, which might be computed from the current state $(Q,\underline{\underline{R}})$. (Whether or not $\underline{R}$ is actually a subdeduction of $\underline{\underline{\mathcal{D}}}$ depends upon another parametric procedure, Subsume, in Ref.)

The definition of E from $E_n,\cdots,E_0,E_\mu$ now reduces to the definition of f from $\{h_n,\cdots,h_0,h_\mu,g_\mu\}$ where E_k is essentially defined by a cost-function g_k and an "additional completion cost estimator" h_k. It suffices to set $g = g_\mu$. The design of h should insofar as possible implement the "hierarchical control structure"

hinted at in Compositions of Enqueuing Functions so as to utilize each search strategy E_k or E_μ to an appropriate degree in its intended application environment.

Remark. It should be clear that the above is a heuristic design strategy for defining E from $E_n,\cdots,E_0,E_\mu$ rather than an outline of some alleged formal definition. The claim that one can design better search strategies by this hierarchical approach rather than by designing E "from scratch" on the basis of $\Delta_M{\cdot}\Delta_\mu$ is at present an unsubstantiated intuition. Thus, the use of an explicit composition $(E_M{\cdot}E_\mu)$ notation is (at the present time) somewhat misleading.

The representation of deductions within a normal refutation procedure based on Ref should optimize retrieval of most-frequently needed information in the operation of Res_Δ and Enq while suppressing irrelevant details. Suppose that $\Delta_M = \Delta_n \cdot \cdots \cdot \Delta_0$, $\Delta_\mu = ND(\mathcal{E}_0,\succ)$ and $\underline{\underline{\mathcal{D}}}$ is in $\Delta = \Delta_M{\cdot}\Delta_\mu$. Then for each clause C in $\mathcal{D}$, the component C_ℓ in (12) and $Level_{\underline{\underline{\mathcal{D}}}}(C)$ should probably be "immediately available", as should the designated literal within C_ℓ. The composite restrictions of Δ_M will require efficient access to the $\mathcal{E}_k$-resolution premises of C $(k=n,\cdots,Level_{\underline{\underline{\mathcal{D}}}}(C))$, and Δ_μ will require efficient access to the equations among the designated literals of $\mathcal{E}_0$-resolution ancestors of C, in addition to the derived equations in $\mathcal{D} \cup \mathcal{E}_0$.

The representation of terms and atoms in Ref can also be optimized for the given initial normal form $NF(\mathcal{E}_0,\succ)$ used by Δ_μ,

assuming that $ND(\mathcal{E}_0,\succ)$ will be used frequently enough to justify "specializing" a proof procedure at this basic level. Normally this is accomplished by incorporating into Ref specialized algorithms and data structures for processing associative or associative and commutative operators; these facilities are invoked for each operator declared to have the pertinent axiomatic properties. Hardware-representations for "integers" or "reals" and hardware-evaluation of constant terms (and atoms) both illustrate this form of proof-procedure specialization [22].

3. COMPLETENESS RESULTS

This chapter derives completeness results for (generalized) $\mathcal{E}$-resolution refinements and their normal compositions. The basic results (including those in the Abstract) are summarized by Theorems 1-10 and their corollaries. Lemmas used in the proofs of these theorems are proved in §A.

In §3.1 it is shown that $HR(\mathcal{E},\succ,s)$ is $\mathcal{E}$-complete, and that hyper-$\underline{\mathcal{E}}$-resolution has a "strong liftability" property whereby ground deductions from a clause-set $\mathcal{C}'$ can be generalized to general deductions from a clause-set $\mathcal{C}$ where each clause of $\mathcal{C}'$ is subsumed by a clause of $\mathcal{C}$.

The completeness of $ND(\mathcal{E},\succ)$ on sets of unit clauses is derived in §3.2, where it is shown that if $\mathcal{E}$ is "closed" under certain operations then $\Longrightarrow^*_{\underline{\mathcal{E}}+}$ is a complete reducibility relation on constant terms. A derivation procedure $\underline{Cl}(\mathcal{E})$, similar to the proof procedure $\underline{Ref}(\mathcal{E})$, is shown to derive a closed reduction from $(\mathcal{E},\succ)$ provided that $\succ$ is an $\mathcal{E}'$-complexity ordering where $\mathcal{E}' \leq \mathcal{E}$ (Theorem 6).

Preservation of strong liftability and completeness under normal compositions is investigated in §3.3. Theorem 9 describes sufficient conditions for a resolution micro-refinement Δ_μ to yield a complete normal refinement $\Delta_M \cdot \Delta_\mu$ when composed with a strongly liftable resolution refinement Δ_M . Theorem 10 (Corollary of Theorem B in Abstract) describes a partial completeness property of normal refinements $\Delta_M \cdot ND(\mathcal{E},\succ)$.

Convention. In order to ensure that $\mathcal{E}^+$ (the set of constant instances of members of $\mathcal{C}$) is not empty below, we assume that $V_F^0 \neq 0$.

The following well known result is used in several of the completeness proofs below:

(Relative) Compactness Theorem. Suppose $\mathcal{C}$ is $\mathcal{E}$-contradictory. Then so is some finite subset of $\mathcal{C}$.

Corollary. Suppose $\mathcal{C}$ is $\mathcal{E}$-inconsistent. Then so is some finite subset of $\mathcal{C}$.

Remark. One simple way to derive the theorem is to construct an (exhaustive) finitely branching $\mathcal{E}$-model tree $\mathcal{M}$ where each vertex M is an $\mathcal{E}$-model whose successors $M_1,\cdots,M_{\nu_m}$ are finite extensions of M such that $M \models_{\mathcal{E}} C_1 \vee \cdots \vee C_{\nu_m}$, where C_j is the conjunction of literals in M_j-M. (The initial vertex of $\mathcal{M}$ is the empty set of literals.) It follows by Konig's Lemma that each branch in $\mathcal{M}$ passes through some failure vertex which falsifies $\mathcal{C}$, and the set of such vertices is finite. Since the set of clauses constructable from literals in failure vertices is also finite, some finite subset of $\mathcal{C}$ is $\mathcal{E}$-contradictory. The Corollary follows from the fact that if $\mathcal{C}$ is $\mathcal{E}$-inconsistent, then $\mathcal{C}^+$ is $\mathcal{E}$-contradictory. Similar model-tree constructions are used by Robinson [65] and others.

3.1 Completeness of $\mathcal{E}$-Resolution Refinements

It is assumed below that $\underline{\mathcal{E}} = (\mathcal{E},\succ,s)$ where

(i) $\mathcal{E}$ is a consistent set of clauses;

(ii) $\succ$ is a substitutive partial ordering on $\mathcal{A}_V$ wherein equations precede other atoms; and

(iii) s is a (negative literal) selection function.

Similar assumptions hold for $\underline{\mathcal{E}}_i = (\mathcal{E}_i, \succ_i, s_i)$.

3.1.1 Horn Systems and Hyper-$\underline{\mathcal{E}}$-Resolution

$\mathcal{E}$ is a Horn system with renaming r provided that no clause of $\mathcal{E}$ contains more than one r-positive literal. If r is the identity renaming then $\mathcal{E}$ is a Horn system.

Horn systems occur in many axiomatizations of mathematical systems. The following result indicates that $HR(\mathcal{E}, \succ, s)$ is an appropriate refinement to use when $\mathcal{E}$ is a Horn system with renaming r_s :

Theorem 1. Suppose that $\mathcal{E}$ is a Horn system with renaming r_s and $\mathcal{C}$ is $\mathcal{E}$-inconsistent. Then $HR(\mathcal{E}, \succ, s)$ contains a general refutation of $\mathcal{C} \cup \mathcal{E}$.

The proof of this theorem, based on Lemmas 1-4, is completed in §3.1.3. Notice that a general refutation of $\mathcal{C} \cup \mathcal{E}$ in $HR(\mathcal{E}, \succ, s)$ has a decomposition into hyper-$\underline{\mathcal{E}}$-resolution inference realizations, and that these hyper-$\underline{\mathcal{E}}$-resolution inferences constitute a general hyper-$\underline{\mathcal{E}}$-resolution refutation (by Proposition 1 in §2).

3.1.2 A Ground Completeness Proof Schema

Many completeness results for liftable $\mathcal{E}$-resolution refinements Δ will be based on the following:

Proposition S. If $\mathcal{C}$ is a finite $\mathcal{E}$-contradictory set of constant clauses then Δ contains a refutation of $\mathcal{C} \cup \mathcal{E}$.

Such a proposition can often be proved by induction on $\kappa(\mathcal{C})$, the excess literal parameter of $\mathcal{C}$, defined by

$$\kappa(\mathcal{C}) =_{df} (\text{number of occurrences of atoms in } \mathcal{C}) - (\text{number of clauses in } \mathcal{C}).$$

Thus, if $\mathcal{C}$ is a set of unit clauses then $\kappa(\mathcal{C}) = 0$. The method of argument is given by the following:

Proof schema (by induction on $k = \kappa(\mathcal{C})$) [4].

Base: Suppose $k = 0$. [Prove Proposition S for case where $\mathcal{C}$ is a set of constant literals.]

Induction step: Suppose $k > 0$ and Proposition 2 holds for all $\mathcal{C}$ such that $\kappa(\mathcal{C}) < k$. Select a clause $A \vee B$ in $\mathcal{C}$ where $A \cap B = 0$ and $A, B \neq 0$. Let $\mathcal{C}' = \mathcal{C} - \{A \vee B\}$ and $\mathcal{C}_A = \mathcal{C}' \cup \{A\}$. Then $\mathcal{C}_A$ is $\mathcal{E}$-contradictory and $\kappa(\mathcal{C}_A) < k$. It follows by induction that Δ contains a refutation $\underline{\underline{\mathcal{D}}}_A(0)$ of $\mathcal{C}_A$.

Let $\underline{\underline{\mathcal{D}}}_{A \vee B}(B')$ be the embedding of $\underline{\underline{\mathcal{D}}}_A(0)$ in $\mathcal{C}$. Then $B' \subseteq B$. If $B' = 0$ then let $\underline{\underline{\mathcal{D}}} = \underline{\underline{\mathcal{D}}}_{A \vee B}(0)$. Otherwise let $\mathcal{C}_{B'} = \mathcal{C}' \cup \{B'\}$, and observe that $\mathcal{C}_{B'}$ is $\mathcal{E}$-contradictory and $\kappa(\mathcal{C}_{B'}) < k$, whence Δ contains a refutation $\underline{\underline{\mathcal{D}}}_{B'}(0)$ of $\mathcal{C}_{B'}$. Let $\underline{\underline{\mathcal{D}}}$ be the result of prefixing $\underline{\underline{\mathcal{D}}}_{A \vee B}(B')$ to $\underline{\underline{\mathcal{D}}}_{B'}(0)$.

Now $\underline{\underline{\mathcal{D}}}$ is a refutation of $\mathcal{C} \cup \mathcal{E}$, and is in Δ provided that $A \vee B$ and A have been chosen "appropriately" for the given refinement Δ. ∎

This proof schema is exemplified by the proofs of Lemma 1 and Lemma 2 in §A.1.

Lemma 1. Suppose that $\mathcal{E} \leq \mathcal{L}_V$ and $\mathcal{C}$ is a finite $\mathcal{E}$-contradictory set of constant clauses. Then there exists a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C} \cup \mathcal{E}$.

The following corollary, useful in the proof of Lemma 2, provides a clear semantical motivation for the use of hyper-$\underline{\mathcal{E}}$-resolution or

$HR(\mathcal{E},\succ,s)$ when $\mathcal{E}$ is a Horn system:

Corollary. Suppose that $\mathcal{E}$ is a Horn system and $\{p_i: i<m\}$ is a minimal $\mathcal{E}$-contradictory set of constant literals. Then $\{p_i: i<m\}$ contains at most one negative literal, and if each literal of $\{p_i: i<m\}$ is positive then $\mathcal{E}^+$ contains a negative clause $\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$ such that $\{p_i: i<m\} \models\!\!\!\models_{\mathcal{E}} q_j$ $(j=0,\cdots,n-1)$.

Proof. It follows by the Compactness Theorem (Corollary) that some minimal (hence finite) subset $\mathcal{C}$ of $\mathcal{E}^+ \cup \{p_i: i<m\}$ is contradictory. It follows by Lemma 1 that there exists a hyper-E-resolution refutation $\underline{\underline{\mathcal{D}}}(0)$ of $\mathcal{C}$, and $\{p_i: i<m\} \subseteq \text{Base}(\underline{\underline{\mathcal{D}}}(0))$ by minimality of $\{p_i: i<m\}$.

Suppose $(\{B_i \vee q_i: i<m\} \cup \{p_j\} \vdash C)$ is a hyper-E-resolution inference where p_j is negative. Then $B_i \vee q_i$ is positive, whence $B_i = 0$ because $\mathcal{C}$ is a Horn system $(i=0,\cdots,n-1)$. Therefore $C = 0$, whence $\underline{\underline{\mathcal{D}}}(0)$ contains at most one such inference and p_j is the only negative literal in $\{p_i: i<m\}$.

Suppose p_i positive $(i=0,\cdots,m-1)$ and let $(\{q'_i: i<m'\}$ $\{(\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1})\} \vdash 0)$ be the final inference in $\underline{\underline{\mathcal{D}}}(0)$. Clearly, $(\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1})$ is in $\mathcal{E}^+$, and $\{p_i: i<m\} \models\!\!\!\models_{\mathcal{E}} q_j$ because $\{p_i: i<m\} \models\!\!\!\models_{\mathcal{E}} \{q'_i: i<m'\}$ and $\{q'_i: i<m'\} \cup \{\tilde{q}_j\}$ is $\mathcal{E}$-contradictory $(j=0,\cdots,m'-1)$.∎

Lemma 2. Suppose that $\mathcal{E}$ is a Horn system with renaming r_s, and $\mathcal{C}$ is a finite $\mathcal{E}$-contradictory set of constant clauses. Then there exists a hyper-$\underline{\underline{\mathcal{E}}}$-resolution refutation of $\mathcal{C} \cup \mathcal{E}$.

3.1.3 Strong Liftability of Hyper-$\mathcal{E}$-Resolution

In view of the fact that the hyper-$\mathcal{E}$-resolution refinement is the class of all deductions composed of hyper-$\mathcal{E}$-resolution inferences, strong liftability (§1.3.8) of the hyper-$\mathcal{E}$-resolution refinement amounts to the following:

Lemma 3. Suppose that

$$(\{B'_0 \vee p'_0,\cdots,B'_{m-1} \vee p'_{n-1},\ B'_m \vee \tilde{q}'_0 \vee \cdots \vee \tilde{q}'_{n-1}\} \vdash C' \qquad (3)$$

is a hyper-$\mathcal{E}$-resolution inference where

$$C' = (B'_0\theta' - p'_0\theta') \vee \cdots \vee (B'_{m-1}\theta' - p'_{m-1}\theta') \vee B'_m\theta'$$

and that $\mathcal{B} = \{B_0 \vee p_0,\cdots,B_{m-1} \vee p_{m-1},\ B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}\}$, a separated set of clauses such that $B_m\eta \subseteq B'_m$, $q_i\eta = q'_i\eta (i=0,\cdots,n-1)$, $B_i\eta \subseteq B'_i$, and $p_i\eta \subseteq p'_i$ $(i=0,\cdots,n-1)$. Let θ be any divisor of $\eta\cdot\theta'$ such that

(i) $\{p_0\theta,\cdots,p_{m-1}\theta\ ,\ (\tilde{q}_0\theta \vee \tilde{q}_{n-1}\theta)\}$ is an $\mathcal{E}$-contradiction; and

(ii) if $p \in B_i$ and $p\eta\theta' = p_i\eta\theta'$ then $p\theta = p_i\theta$ $(i=0,\cdots,m-1)$.

Then the inference

$$\{B_0 \vee p_0,\cdots,B_{m-1} \vee p_{m-1},\ B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}\} \vdash C \qquad (4)$$

where $C = (B_0\theta - p_0\theta) \vee \cdots \vee (B_{m-1}\theta - p_{m-1}\theta) \vee B_m\theta$ is a hyper-$\mathcal{E}$-resolution inference, and $C[\eta\cdot\theta'/\theta] \subseteq C'$.

Corollary. Suppose that $\mathcal{E}$ is a Horn system with renaming r_s and $\mathcal{B}$ is an $\mathcal{E}$-inconsistent set of clauses. Then there exists a general hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{B} \cup \mathcal{E}$.

Proof. $\mathcal{B}^+$ is $\mathcal{E}$-contradictory. By Relative Compactness there exists a finite $\mathcal{E}$-contradictory subset $\mathcal{C}$ of $\mathcal{B}^+$. By Lemma 2 there exists a hyper-$\underline{\mathcal{E}}$-resolution refutation $\underline{\mathfrak{D}}'(0)$ of $\mathcal{C} \cup \mathcal{E}$. Let $\underline{\mathfrak{D}}(0)$ be a hyper-$\underline{\mathcal{E}}$-resolution lifting of $\underline{\mathfrak{D}}'(0)$ based on some branch mapping π (§1.3.8). It follows by Lemma 3 that $\underline{\mathfrak{D}}(0)$ is a general hyper-$\underline{\mathcal{E}}$-resolution refutation, and $\mathrm{Base}(\underline{\mathfrak{D}}(0)) \leq (\mathcal{B} \cup \mathcal{E})^{\sim}$. ∎

$\mathrm{HR}(\mathcal{E}, \succ, s)$ is not in general strongly liftable; s may select negative literals from $(B_m\eta \vee \tilde{q}_0\eta \vee \cdots \vee \tilde{q}_{n-1}\eta)$ in an entirely different order than that in which it selects negative literals from $B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$. However, the following lemma shows that if we include a sufficient number of variants among the premises of a hyper-$\underline{\mathcal{E}}$-resolution inference (4), then (4) can be realized by an $\underline{\mathcal{E}}$-resolution deduction containing n inferences:

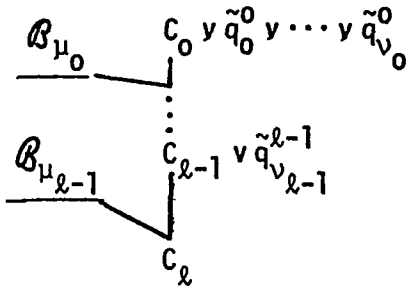
Lemma 4. Suppose that (4) is a hyper-$\underline{\mathcal{E}}$-resolution inference with separated premises $\mathcal{B} = \mathcal{B}_0 \cup \cdots \cup \mathcal{B}_{n-1} \vee \{B_m \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}\}$ where

(i) $\mathcal{B}_i = \{B_k^i \vee p_k^i : k < m_i\} \subseteq \{B_j \vee p_j : j < m\}$ $(i=0,\cdots,n-1)$;

(ii) $\{p_k^i\theta : k < m_i\} \cup \{\tilde{q}_i\theta\}$ is an $\mathcal{E}$-contradiction $(i=0,\cdots,n-1)$; and

(iii) if $i \neq j$ then $\mathcal{B}_i \wedge \mathcal{B}_j = 0$ $(i,j < n)$.

Then (4) is realized by an $\underline{\mathcal{E}}$-resolution deduction $\underline{\mathfrak{D}}(C)$:



(5)

where

(iv) $C^0 = B_m$ and $\{q^0_0,\cdots,q^0_{\nu_0}\} = \{q_0,\cdots,q_{n-1}\}$;

(v) $s(C_i \vee \tilde{q}^i_0 \vee \cdots \vee \tilde{q}^i_{\nu_i}) = \tilde{q}^i_{\nu_i}$;

(vi) $C_{i+1} = (B^j_0\theta_i - p_0\theta_i) \vee \cdots \vee (B_{m_{j-1}}\theta_i - p_{m_{j-1}}\theta_i) \vee C_i\theta_i$

where $j = \mu_i$,

$\theta_i = [x\theta\ /x\colon x$ occurs in $\mathcal{B}_j \cup \{q_j\}]$; and

(vii) $\{q^{i+1}_0,\cdots,q^{i+1}_{\nu_{i+1}}\} = \{q^i_0\theta_i,\cdots,q^i_{\nu_i-1}\theta_i\} - \{q^i_{\nu_i}\theta_i\}$.

Corollary. For each general hyper-$\mathcal{E}$-resolution deduction $\mathfrak{D}'$, HR($\mathcal{E}$,$\succ$,s) contains a general deduction $\mathfrak{D}$ such that Base($\mathfrak{D}$) $\subseteq$ Base($\mathfrak{D}'$)$^\sim$ and $\mathfrak{D}$ has the same conclusions as $\mathfrak{D}'$.

Indication of proof. Let $\mathfrak{D}$ be the deduction in HR($\mathcal{E}$,$\succ$,s) having decomposition $\{\mathfrak{D}_i(C_i)\colon i<n\}$ where $\mathfrak{D}_i(C_i)$ is a realization of a hyper-$\mathcal{E}$-resolution inference $(\mathcal{B}_i \vdash C_i)$ (obtained as in Lemma 4) and $\{\mathcal{B}_i \vdash C_i\colon i<n\}$ is the set of inferences in $\mathfrak{D}'$.

Proof of Theorem 1. By Lemma 3 (Corollary) there exists a general hyper-$\mathcal{E}$-resolution refutation $\mathfrak{D}'(0)$ of $\mathcal{B} \cup \mathcal{E}$. It follows by Lemma 4 (Corollary) that HR($\mathcal{E}$,$\succ$,s) contains general refutation $\mathfrak{D}(0)$

of $\mathcal{B} \cup \mathcal{E}$.∎

3.2 Completeness of Resolution Micro-Refinements

It is assumed below that $\underline{\mathcal{E}}$ is a reduction system $(\mathcal{E}, \succ)$ where $\mathcal{E}$ is a set of clauses.

3.2.1 Unit Completeness of ND$(\mathcal{E}, \succ)$

A refinement Δ is unit $\mathcal{E}$-complete provided that if $\mathcal{B}$ is an $\mathcal{E}$-inconsistent set of unit clauses (literals) then Δ contains a refutation of $\mathcal{B} \cup \mathcal{E} \cup [x=x]$.

Theorem 2. Suppose $\succ$ is an $\mathcal{E}'$-complexity ordering where $\mathcal{E}' \subseteq \mathcal{E}$. Then ND$(\mathcal{E}, \succ)$ is unit $\mathcal{E}$-complete.

This result is a trivial corollary of Theorem 3, which also describes a restricted "lifting property" of ND$(\mathcal{E}, \succ)$. Define $[t]_{\mathcal{E}}$ for t in $\mathcal{T}_V$ by

$$[t]_{\mathcal{E}} =_{df} \{s: s =_{\mathcal{E}} t\}$$

Theorem 3. Suppose that p shares no variables with q and $p\theta \models|_{\mathcal{E}} q\theta$ where $x\theta$ is the first constant term in $[x\theta]_{\mathcal{E}}$ (with respect to $\succ$) for each variable x occurring in p or q . Then ND$(\mathcal{E}, \succ)$ contains a general refutation $\underline{\mathcal{D}}(0)$ of $\{p, \tilde{q}\}$ such that

$$[x\sigma_{\underline{\mathcal{D}}(0)}/x: \text{ x occurs in p or q }] \text{ divides } \theta .$$

This result follows from Theorem 5, Lemma 9 (Corollary 2), and Theorem 6 (§3.2.3). Lemma 9 covers the case where $\underline{\mathcal{E}}$ is a closed reduction system (§3.2.2). Theorem 6 shows that given any reduction system $(\mathcal{E}, \succ)$ where $\succ$ is an $\mathcal{E}'$ complexity order and $\mathcal{E}' \subseteq \mathcal{E}$, ND$(\mathcal{E}, \succ)^-$ contains

a general deduction of a set $Cl(\mathcal{E})$ from $\mathcal{E}$ such that $(Cl(\mathcal{E}),\succ)$ is closed.

Corollary. Suppose $A \vee p$ shares no variables with $B \vee \tilde{q}$ and $p\theta \models_{\mathcal{E}} q\theta$ where $x\theta$ is the first constant term in $[x\theta]_{\mathcal{E}}$ for each variable x occurring in $(A \vee p)$ or $(B \vee \tilde{q})$. Then $ND(\mathcal{E},\succ)$ contains a general realization $\underline{\underline{\mathcal{D}}}$ for an $\mathcal{E}$-resolution inference

$$\{A \vee p,\ B \vee \tilde{q}\} \vdash (A\sigma_{\underline{\underline{\mathcal{D}}}} - p\sigma_{\underline{\underline{\mathcal{D}}}}) \vee (B\sigma_{\underline{\underline{\mathcal{D}}}} - \tilde{q}\sigma_{\underline{\underline{\mathcal{D}}}})$$

where $[x\sigma_{\underline{\underline{\mathcal{D}}}}/x$: x occurs in $A \vee p$ or $B \vee q]$ divides θ.

Proof. Obtain $\underline{\underline{\mathcal{D}}}(C)$ by embedding a general refutation $\underline{\underline{\mathcal{D}}}'(0)$ for $(\mathcal{E} \cup [x=x])^{\sim} \cup \{p,\tilde{q}\}$ in $(\mathcal{E} \cup [x=x])^{\sim} \cup \{A \vee \underline{p},\ B \vee \underline{q}\}$, where $\underline{\underline{\mathcal{D}}}'(0) \in ND(\mathcal{E},\succ)$.

Remark. The corollary is essentially a "lifting lemma" for constant deductions realizing resolution inferences

$$\mathcal{A}' \cup \{(A \vee p)\theta,(B \vee \tilde{q})\theta\} \vdash (A\theta - p\theta) \vee (B\theta - q\theta) \qquad (6)$$

where p, q, and θ are as above, $\{p' \in A \vee p$: $p'\sigma_{\underline{\underline{\mathcal{D}}}} = p\sigma_{\underline{\underline{\mathcal{D}}}}\}$ $= \{p' \in A \vee p$: $p'\theta = p\theta\}$, and $\{p' \in B \vee \tilde{q}$: $p'\sigma_{\underline{\underline{\mathcal{D}}}} = \tilde{q}\sigma_{\underline{\underline{\mathcal{D}}}}\} = \{p' \in B \vee \tilde{q}$: $p'\theta = \tilde{q}\sigma_{\underline{\underline{\mathcal{D}}}}\}$.

3.2.2 Completeness Properties of Closed Reduction Systems

Unit $\mathcal{E}$-completeness of $ND(\mathcal{E},\succ)$ intuitively involves the completeness of some reduction system $(\mathcal{E}',\succ)$ where $\mathcal{E}'$ is derived from $\mathcal{E}$: if $p\theta =_{\mathcal{E}} q\theta$ then $p\theta \Rightarrow^*_{\underline{\mathcal{E}}'} q'$ and $q\theta \Rightarrow^*_{\underline{\mathcal{E}}'} q'$. Under certain circumstances, it follows that $p \Rightarrow_{\underline{\mathcal{E}}'} p'$ and $q \Rightarrow_{\underline{\mathcal{E}}'} p'$ where p' subsumes q'. The following theorem is useful in the analysis of

a specific refinement $ND(\mathcal{E},\succ_D)$ in §C and §D :

Theorem 4. Suppose that $\underline{\mathcal{E}'}$ satisfies (i)-(v):

(i) $\mathcal{E}'$ is a finite set of equations [s=t] where each variable in t is also in s.

(ii) $\succeq$ is an invariant partial ordering on $\mathcal{T}_V$.

(iii) Every descending chain in $\succeq$ is finite.

(iv) $s \succ t$ for each equation [s=t] in $\mathcal{E}'$.

(v) For each separated pair $\{[s=t],[u[r]=v]\} \subseteq \mathcal{E}'^{\cup}$ such that $r \notin V_I$ and $mgu\{r,s\} = \theta \in \Sigma_V$, there exists a term v' such that $u[t]\theta \rightarrow^*_{\underline{\mathcal{E}'}} v'$ and $v\theta \rightarrow^*_{\underline{\mathcal{E}'}} v'$.

Then $\underline{\mathcal{E}'}$ is normally complete and canonical. That is, for each pair of terms r,s such that $r =_{\mathcal{E}'} s$, $NF(\mathcal{E}',\succ)$ contains a unique term t such that $r \Rightarrow^*_{\underline{\mathcal{E}'}} t$ and $s \Rightarrow^*_{\underline{\mathcal{E}'}} t$.

This result is derived in §C.4. In cases where we cannot derive a system $\underline{\mathcal{E}'}$ satisfying (i)-(v) from $\underline{\mathcal{E}}$, we can nevertheless obtain somewhat weaker completeness properties (expressed in terms of $\Rightarrow^*_{\mathcal{E}'}$) by requiring that $\mathcal{E}'$ be closed under certain operations such as normal replacement inferences and reductions of equations to normal form with respect to all other equations.

Closed reduction systems. Specifically, we say that $\underline{\mathcal{E}}$ is closed provided that it satisfies (i)-(iii) for each equation [u=v] in $\mathcal{E}$:

(i) If $u \neq v$ and $[u=v] \in NF(\mathcal{E},\succ)$ then $[v=u] \in \mathcal{E}^*$.

(ii) If $[u=v] \notin S[s=t]$ for all $[s=t]$ in $\mathcal{E} - [u=v]^{\cup}$ and $[u=v] \notin [x=x]^*$ then $[u=v] \rightarrow^*_{\underline{\mathcal{E}}} [u'=v'] \in NF(\mathcal{E},\succ)$ where

$(\mathcal{E} \cup [x=x])^*$ contains $[u'=v']$ or $[v'=u']$.

(iii) Suppose $\mathcal{E}$ contains $[s=t]$ such that $(\{[s=t]\eta, [u'[r]=v]\} \vdash [u'[t\eta]=v]\theta)$ is an $Rp_{\mathcal{E}}$-inference where $u = u'[r]$ and η is a simplest substitution such that $[s=t]\eta$ shares no variables with $[u=v]$. Then $[u'[t\eta]=v]\theta \rightarrow^*_{\mathcal{E}} [u''=v'']$ where $(\mathcal{E} \cup [x=x])^*$ contains either $[u''=v'']$ or $[v''=u'']$.

In order to make up for the fact that (iv) in Theorem 4 does not hold in general, we work with $\mathcal{E}^+$, the set of constant equations in $\mathcal{E}^*$, instead of $\mathcal{E}$:

Theorem 5. Suppose $\mathcal{E}$ is a closed reduction system. Then for each constant term u, $u \Rightarrow^*_{\mathcal{E}^+} v$ where v is the first constant term of $[u]_{\mathcal{E}}$ (according to $\succ$).

The proof of this theorem is based on Lemmas 5-8 below.

Lemma 5. Suppose u and v are constant terms such that $u =_{\mathcal{E}} v$. Then $u =_{\mathcal{E}^+} v$.

Define $\hat{\mathcal{E}}$ by

$\hat{\mathcal{E}} =_{df} \{[s'=t'] \in \mathcal{E} :$ If $\mathcal{E}^{\sim}$ contains $[s=t]$ such that $s'=s\eta$ for some simplest η then $t' \not\succ t\eta\}$.

Lemma 6. For each term u there exists a unique term $\hat{u}$, such that $u \Rightarrow^*_{\hat{\mathcal{E}}} \hat{u} \in NF(\mathcal{E}, \succ)$.

$\hat{u} =_{df}$ the unique term in $NF(\mathcal{E}, \succ)$ such that $u \Rightarrow^*_{\hat{\mathcal{E}}^+} \hat{u}$.

$\bar{u} =_{df}$ a minimal term in $[u]_{\mathcal{E}}$ with respect to $\succ$.

Let $\mathcal{J}_\nu = \{s_\alpha : \alpha < \nu\}$, an enumeration of the constant terms in $\mathcal{J}_\nu$ such that if $\beta > \alpha$ then $s_\beta \succ s_\alpha$. (Recall that $\succ$ well-orders constant terms.) Define $\mathcal{E}_\beta$ for each ordinal number $\beta < \nu$ by

$$\mathcal{E}_\beta =_{df} \{[s_\alpha = t'] \in \mathcal{E}^+ : s_\beta \succ s_\alpha \succ t'\}$$

Lemma 7. Suppose $u =_{\mathcal{E}_\beta} v$ where $\hat{s} = \hat{t}$ for each equation $[s=t]$ in $\mathcal{E}_\beta$. Then $\hat{u} = \hat{v}$.

Corollary. Suppose $\hat{s} = \hat{t}$ for each equation $[s=t]$ in $\mathcal{E}^+$. Then $\hat{u} = \overline{u}$ $(u \in \mathcal{T}_\nu)$.

Proof. Suppose $u \in \mathcal{T}_\nu$. Then $u =_{\mathcal{E}_\beta} \overline{u}$ for some $\beta < \nu$. It follows by Lemma 7 that $\hat{u} = \hat{\overline{u}} = \overline{u}$. ∎

Lemma 8. Suppose $\mathcal{E}$ is closed. Then $\hat{s} = \hat{t}$ for each equation $[s=t]$ in $\mathcal{E}^+$.

Theorem 5 follows from Lemma 8 and the preceding corollary.

Lemma 9. Suppose that $\mathcal{E}$ is a closed reduction system, $u\theta$ is a constant term, and $\theta \in \Sigma_{\mathcal{E}}$. Then there exists an $\mathcal{E}$-normal derivation $\mathcal{D}$ from $\mathcal{E} \cup \{Pu\}$ to (Pv) where (Pv) subsumes $(P\overline{u\theta})$ and $[x\sigma_{\mathcal{D}}/x: x \text{ occurs in } u]$ divides θ.

Corollary 1. Suppose that $\mathcal{E}$ is closed, $\mathcal{E}$ consists entirely of equations, $p\theta$ is a constant literal, and $\theta \in \Sigma_{\mathcal{E}}$. Then there exists a general $\mathcal{E}$-normal derivation $\mathcal{D}$ from $\mathcal{E} \cup \{p\}$ to q where q subsumes $\overline{p\theta}$, the first constant literal of $\{q': q' \vdash_{\mathcal{E}} p\theta\}$; moreover, $[x\sigma_{\mathcal{D}}/x: x \text{ occurs in } p]$ divides θ.

Proof. It suffices to prove this corollary for the case where $Qu_1 \cdots u_n$, an atom. Then $\overline{p\theta} = (Q(\overline{u_1\theta}) \cdots (u_n\theta))$. It follows by the lemma that there exists a general $\mathcal{E}$-normal derivation $\mathcal{D}_1$ from $\mathcal{E}^\omega \cup (Qu_1 \cdots u_n)$ to $(Qv_1(u_2\sigma_1)\cdots(u_n\sigma_1))$ subsuming

$Q(\overline{u_1\theta})(u_2\theta)\cdots(u_n\theta)$ where $\sigma_1 = \sigma_{\underline{\mathfrak{D}}_1}$ and $[x\sigma_{\underline{\mathfrak{D}}_1}/x$: x occurs in p] divides θ . Proceeding by induction, we eventually derive $(Qv_1\cdots v_n)$ subsuming $(Q\overline{u_1\theta}\cdots\overline{u_n\theta})$ by means of an $\underline{\mathcal{E}}$-normal derivation $\underline{\mathfrak{D}}_n$ where $[x\sigma_{\underline{\mathfrak{D}}_n}/x$: x occurs in p] divides θ .∎

Corollary 2. Suppose that $\underline{\mathcal{E}}$ is closed, $\mathcal{E}$ consists entirely of equations, p and q share no variables, $\theta \in \Sigma_{\underline{\mathcal{E}}}$, and $p\theta, q\theta$ are constant literals such that $p\theta \models_{\mathcal{E}} q\theta$. Then $ND(\mathcal{E},\succ)$ contains a general refutation $\underline{\mathfrak{D}}$ of $\mathcal{E}^{\sim} \cup \{p,\tilde{q}\}$, and $[x\sigma_{\underline{\mathfrak{D}}}/x$: x occurs in p or in q] divides θ .

Proof. Observe that $\overline{p\theta} = \overline{q\theta}$. Corollary 1 implies the existence of general $\underline{\mathcal{E}}$-normal derivations $\underline{\mathfrak{D}}_1$ from p to p' subsuming $\overline{p\theta}$, and $\underline{\mathfrak{D}}_2$ from q to q' subsuming $\overline{q\theta}$, such that $\underline{\mathfrak{D}}_1$ and $\underline{\mathfrak{D}}_2$ share no variables and σ' divides θ , where

$$\sigma' =_{df} [x\sigma_{\underline{\mathfrak{D}}_1}\cdot\sigma_{\underline{\mathfrak{D}}_2}/x:\ x \text{ occurs in p or in q}]$$

It follows that $p'[\theta/\sigma'] = q[\theta/\sigma']$. Let $\underline{\mathfrak{D}}$ be the result of suffixing the $Cut(\mathcal{E},\succ)$-inference

$$\{p',\tilde{q}'\} \vdash 0$$

to $\underline{\mathfrak{D}}_1(p')$ and $\underline{\mathfrak{D}}_2(\tilde{q}')$. Clearly $[x\sigma_{\underline{\mathfrak{D}}}/x$: x occurs in p or in q] divides θ , and $\underline{\mathfrak{D}}$ is in $ND(\mathcal{E},\succ)$.∎

3.2.3 Convergence of an $\underline{\mathcal{E}}$-normal Derivation Procedure

The procedure Cl below is a modification of Ref (§1.3.6) which, given a set $\mathcal{E}$ of equations, defines a composite $\underline{\mathcal{E}}$-normal deduction $\underline{Cl}(\mathcal{E})$, where $\underline{\mathcal{E}} = (\mathcal{E},\succ)$ for some $\mathcal{E}'$-complexity ordering $\succ$ such that $\mathcal{E}' \subseteq \mathcal{E}$. We show below that Cl , applied to $\mathcal{E}$,

"converges" to a set $Cl(\mathcal{E})$ such that $(Cl(\mathcal{E}),\succ)$ is a closed reduction system.

Define $Par(\mathcal{E},\succ)$ by

$$Par(\mathcal{E},\succ) =_{df} (\mathcal{E} \cup \{[u[t\eta] = v]\theta:\ \mathcal{E} \text{ contains } [u[r]=v]\ ,$$
$$[s=t] \text{ such that } v \not\succ u[r]\ ,$$
$$[u[r]=v] \in NF(\mathcal{E} - [u[r]=v]^{\sim},\succ) \text{ and}$$
$$(\{[s=t]\eta,[u[r]=v]\} \vdash [u[t\eta]=v]\theta)$$
$$\text{is an } Rp_{(\mathcal{E},\succ)}\text{-inference, where } \eta \text{ is a standard}$$
$$\text{simplest substitution such that } [s=t]\eta \text{ shares no}$$
$$\text{variables with } [u[r]=v] \text{ and } \theta = mgu\{r,s\eta\}\}.$$

Let $\prec_{\mathcal{E}}$ be the partial ordering of $Par(\mathcal{E},\succ)$ such that $[s=t] \prec_{\mathcal{E}} [u[t\eta]=v]\theta$ and $[u(r)=v] \prec_{\mathcal{E}} [u[t\eta]=v]\theta$ for each $Rp_{\underline{\underline{\mathcal{E}}}}$-inference $(\{[s=t]\eta,\ [u[r]=v]\} \vdash [u[t\eta]=v]\theta)$ used in generating an equation in $Par(\mathcal{E},\succ) - \mathcal{E}$.

Define $\rightarrow$ on equations by

$$[\overrightarrow{u=v}] =_{df} \begin{cases} 0\ , & \text{if } u=v\ ; \\ \{[u=v]\}, & \text{if } u \succ v\ ; \\ \{[v=u]\}, & \text{if } v \succ u\ ; \text{ and} \\ \{[u=v],[v=u]\}\ , & \text{otherwise.} \end{cases}$$

Define $\underline{\underline{Cl}}(\mathcal{E})$ by

$Cl(\mathcal{E}) =_{df}$ [Q: = Enq(nil,Par($\mathcal{E}$,$\succ$)) ;

$\underline{R}$: = ($\mathcal{E}$, $\prec_{\mathcal{E}}$) ;

Result: If Q = $\underline{nil}$

then $\underline{R}$

else [Next(A,Q) ;

Subsume(A,$\underline{R}$) ;

Res_{Δ}(A,$\underline{R}$,T) ;

Q: = Enq(Q,T) ;

Result]].

Notes on Operation

1. The operation of $\underline{Cl}(\mathcal{E})$ is basically like the operation of $\underline{Ref}(\mathcal{E})$ (§1.3.6) except for differences in Next, Subsume, and Res_{Δ} noted below.

2. On the kth iteration of Result (starting with k = 0), Next(A,Q) deletes $p_k = Q_k(0)$ from Q and sets $A = q_k$ where $p_k \Rightarrow^*_{\underline{R'}_k} q_k \in NF(R'_k,)$ and $R'_k = R_k - \vec{p}_k$.

3. Subsume(A,$\underline{R}$) sets $\underline{R} = R_{k+1} = R_k \cup \vec{q}_k$. For formal analysis purposes we do not actually delete subsumed equations; we merely restrict their role in subsequent inferences, in accordance with ND($\mathcal{E}$,$\succ$).

4. On the kth iteration, $\underline{Res}_{\Delta}$(A,$\underline{R}$,T) sets $\underline{T} = (U_k \cup V_k)$

U_k = {[u[tη]θ = vθ]: $R_{k+1} \supseteq$ {[s=t],[u[r]= v]}

where $(\vec{q}_k - R^*_k) \cap$ {[s=t],[u[r]= v]} $\neq 0$,

$v \not\succ u[r]$, and ({[s=t]η,[u[r]= v]} $\vdash$ [u[tη]= v]θ

is an $Rp_{\underline{R}_k}$-inference, where η is a standard simplest substitution such that [s=t] shares no variables with

$[u[r] = v]$ and $\theta = mgu\{r,s\eta\}$

$V_k = \{[u=v] \in R_k \cap NF(R_k,\succ)$: $[u=v] \notin S([s=t])$ for all $[s=t]$ in $(R_k - [u=v]^{\sim})$, and $\vec{q}_k$ contains $[s=t]$ such that $r = s\theta \succ t\theta$ where r occurs in $[u=v]$ and θ is a simplest substitution such that $r = s\theta\}$.

U_k is the set of new normal Rp inferences based on a member of $\vec{q}_k$, and V_k is the set of equations in R_k which have become reducible as a result of including $\vec{q}_k$ in R_{k+1}. $\underline{Res}_{\Delta}(A,\underline{R},T)$ also extends $\prec_{\underline{R}}$, recording the new inference with conclusions in $T - R_k$.

The following result completes the proof of Theorem 3.

Theorem 6. Suppose $\succ$ is an $\mathcal{E}'$-complexity ordering where $\mathcal{E}' \subseteq \mathcal{E}$. Then $\underline{Cl}(\mathcal{E})$ is a composite $\underline{\mathcal{E}}$-normal deduction $(Cl(\mathcal{E}),\succ)$ such that $(Cl(\mathcal{E}),\succ)$ is closed and $\mathcal{E} \subseteq Cl(\mathcal{E})$.

Proof. If $Q_{t+1} = \underline{nil}$ for some first $t \in N$ then set $p_{t+k} = p_t$ and $q_{t+k} = q_t$ $(k \in N)$. Then $Cl(\mathcal{E}) = \mathcal{E} \cup \bigcup\{\vec{q}_k$: $k \in N\}$ and $R_k = \mathcal{E} \cup \bigcup\{\vec{q}_i$: $i < k\}$. Let $\underline{\mathcal{D}}_k$ be the composite $\underline{\mathcal{E}}$-normal deduction with inferences all of the form $\{p_k,q\} \vdash q'$ where $q \in R_k'$. Each such inference is a $CRP_{\underline{R}_k}$-inference. It is easily verified that $\underline{Cl}(\mathcal{E})$, with decomposition $\{\underline{\mathcal{D}}_k$: $k \in N\}$ (finite or infinite) is a composite $\underline{\mathcal{E}}$-normal derivation. It suffices to prove that $\underline{Cl}(\mathcal{E})$ satisfies conditions (i)-(iii) in the definition of closed with $\underline{Cl(\mathcal{E})}$ for $\underline{\mathcal{E}}$. Let $[u=v]$ be an equation in $Cl(\mathcal{E})$.

(i) Suppose $u \neq v$ and $[u=v] \in NF(Cl(\mathcal{E}),\succ)$. In view of the facts that $\mathcal{E} \subseteq Range(Q_0)$ and Enq is a fair enqueuing function, it follows that $[u=v] \in \vec{q}_k$ for some

$k > 0$. Therefore $[v=u] \in \vec{q}_k \subseteq Cl(\mathcal{E})$ also.

(ii) Suppose $[u=v] \notin NF(Cl(\mathcal{E}),)$ and $[u=v] \notin S[s=t]$ for all $[s=t]$ in $Cl(\mathcal{E}) - [u=v]^{\sim}$. Due to the fact that $\succ$ satisfies D.C.C., there exists a maximal increasing sequence $(j_k: k \leq n)$ such that $p_{j_o} = [u=v]$ and $\vec{q}_{j_k} \cap \vec{p}^{*}_{j_{k+1}} \neq 0$. Let θ_k be a simplest substitution such that $\vec{q}_{j_k} \cap \vec{p}_{j_{k+1}} \theta_k \neq 0$ $(k=0,\cdots,n-1)$. It follows that $[u=v] \rightarrow^{*}_{\underline{Cl}(\mathcal{E})} [u'=v'] \in NF(Cl(\mathcal{E}),\succ)$, where either $[u'=v']$ or $[v'=u']$ is $q_{j_n} \theta_{n-1} \cdot \cdots \cdot \theta_o$. Thus, $Cl(\mathcal{E})^*$ contains $[u'=v']$ or $[v'=u']$.

(iii) Suppose $Cl(\mathcal{E})$ contains $[s=t]$ such that $(\{[s=t]\eta, [u'[r]=v]\} \vdash [u'[t\eta]=v]\theta)$ is an $Rp_{\underline{Cl}(\mathcal{E})}$ inference where $u = u'[r]$ and η is a simplest substitution such that $[s=t]\eta$ shares no variables with $[u=v]$.

Case 1: Neither $[s=t]$ nor $[u=v]$ is in $\cup\{\vec{q}_i: i \in N\}$. Then $[s=t]$ and $[u=v]$ are both in $\mathcal{E}$, whence $[u'[t\eta]=v]\theta$ is in $Q_o = Enq(nil, Par(\mathcal{E},\succ))$. Therefore $p_k = [u'[t\ ]=v]\theta$ for some $k \in N$ (by fairness of Enq), whence $[u'[t\eta]=v]\theta \rightarrow^{*}_{\underline{Cl}(\mathcal{E})} [u''=v''] \in NF(Cl(\mathcal{E}),\succ)$, where either $[u''=v''] \in Cl(\mathcal{E})^*$ or $[v''=u''] \in Cl(\mathcal{E})^*$ (using (ii)).

Case 2: One of $[s=t]$, $[u=v]$ is in $\{\vec{q}_i: i \in N\}$. Then there exists a number k such that $R_{k+1} \supseteq \{[s=t],[u'[r]=v]\}$ and $(\vec{q}_k - R^{*}_k) \cap \{[s=t],[u'[r]=v]\} \neq 0$, whence $[u'[t\eta]=v]\theta \in U_k$ by Note 4 above. The conclusion follows as in Case 1.∎

Remark. It can be shown that $(Cl(\mathcal{E}),\succ)$ remains a closed reduction system even if we delete subsumed and reducible equations from R_{k+1} at each step of the computation (by means of a modified Subsume(A,R) operation). While this is a useful thing to know (or to verify) where computational efficiency is important, the incorporation of deletion operations based on $ND(\mathcal{E},\succ)$ into Subsume(A,R) makes the proof of Theorem 6 appreciably more difficult.

3.3 Properties Preserved under Normal Composition

Given a generalized $\mathcal{E}$-resolution refinement Δ and a generalized $\mathcal{E}'$-resolution refinement Δ' where $\mathcal{E}' \subseteq \mathcal{E}$, we may find it useful to derive formal properties or expected performance estimates of the normal composition $\Delta\cdot\Delta'$ as a function of formal properties or expected performance estimates of $\Delta\cdot\Delta'$. The results stated below pertain to liftability and completeness properties necessary for the proofs of Theorem A and of Theorem B (Corollary) in the Abstract.

Strong liftability is a useful property in the results stated below. Since $HR(\mathcal{E},\succ,s)$ is not strongly liftable, as noted in §3.1.3, we define $\overline{HR}(\mathcal{E},\succ,s)$ by

$$\overline{HR}(\mathcal{E},\succ,s) =_{df} \{\underline{\mathcal{D}}: \text{Each inference in } \underline{\mathcal{D}} \text{ is an } (\mathcal{E},\succ,s')\text{-resolution inference, for some selection function } s' \text{ such that } r_{s'} = r_s\}\ .$$

The following lemma is easily unified on the basis of Lemma 4.

Lemma 10. $\overline{HR}(\mathcal{E},\succ,s)$ is strongly liftable.

The following lemma holds by a similar argument:

Lemma 11. For each general deduction $\underline{\mathcal{D}}'$ in $\overline{HR}(\mathcal{E},\succ,s)$, there exists a general deduction $\underline{\mathcal{D}}$ in $HR(\mathcal{E},\succ,s)$ such that $Base(\underline{\mathcal{D}}) = Base(\underline{\mathcal{D}}')$ and $\underline{\mathcal{D}}$ has the same r_s-positive or empty conclusions as $\underline{\mathcal{D}}'$.

Theorem 7. Suppose that Δ is a strongly liftable $\mathcal{E}$-resolution refinement and Δ' is a strongly liftable $\mathcal{E}'$-resolution refinement where $\mathcal{E} \supseteq \mathcal{E}'$. Then the normal composition $\Delta\cdot\Delta'$ is a strongly liftable $\mathcal{E}'$-resolution refinement.

Theorem 7 (proved below) has the following simple corollaries:

Corollary 1. Suppose $\Delta, \mathcal{E}, \mathcal{E}'$ as in Theorem 7. Then $\Delta\cdot\overline{HR}(\mathcal{E}',\succ',s')$ is a strongly liftable $\mathcal{E}'$-resolution refinement.

Corollary 2. If $\mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ where $\mathcal{E}_n \cap \mathcal{L}_V \subseteq \mathcal{E}_0$ then $\overline{HR}(\mathcal{E}_n,\succ_n,s_n)\cdot\ldots\cdot\overline{HR}(\mathcal{E}_0,\succ_0,s_0)$ is a strongly liftable $\mathcal{E}_0$-resolution refinement.

Proof of Theorem 7. First we should verify that the $\mathcal{E}$-resolution calculus is liftable. Given an $\mathcal{E}$-resolution inference $\{B_i': i<n\} \vdash C'$ with kernel $\{C_i: i<n\}$ and induced substitution θ', and a separated set $\{B_i: i<n\}$ where B_i subsumes B_i' but not C', let η be a simplest substitution such that $B_i\eta \subseteq B_i'$ $(i<n)$. Let $C_i = \{p \in B_i: p\eta \in C_i'\}$. Then given any divisor θ of $\eta\cdot\theta'$ such that $\{C_i\theta: i<n\}$ is a set of unit clauses and $\{C_i\theta: i<n\} \not\models_{\mathcal{E}} 0$, we have an $\mathcal{E}$-resolution inference $(\{B_i: i<n\} \vdash C)$ where $C = (B_0 - C_0)\theta \vee \cdots \vee (B_{n-1} - C_{n-1})\theta$. This is a lifting of $\{B_i': i<n\} \vdash C'$ because $C[\eta\cdot\theta'/\theta] \subseteq C'$. The $\mathcal{E}'$-resolution calculus is similarly liftable.

Each ground refutation in $\Delta\cdot\Delta'$ has a decomposition $\{\underline{\mathcal{D}}'_i(C'_i): i<m\}$ into $\mathcal{E}'$-resolution realizations of $\mathcal{E}$-resolution inferences $(\mathcal{B}'_i \vdash C'_i)$ where $\mathcal{B}'_i = \text{Base}(\underline{\mathcal{D}}'_i(C'_i))$. By strong liftability of Δ it suffices to show that if $\underline{\mathcal{D}}'(C')$ is a ground $\mathcal{E}'$-realization of a (ground) $\mathcal{E}$-resolution inference $(\{B'_i: i<n\} \vdash C')$ with kernel $\{C'_i: i<n\}$, $\underline{\mathcal{D}}'(C')$ is the embedding in $\{B'_i: i<n\} \cup \mathcal{E}^*$ of a refutation $\underline{\mathcal{D}}''(0)$ of $\{C'_i: i<n\} \cup \mathcal{E}^*$ where $\underline{\mathcal{D}}''(0) \in \Delta'$, π is a branch mapping for $(\underline{\mathcal{D}}'(C'),\mathcal{B})$ where $\mathcal{B} \leq \{B_k: k<m\} \cup \mathcal{E}^{\cup}$, $B_k\eta \subseteq B'_{jk}$, and $C_k = \{p \in B_k: p\eta \in C'_{jk}\}$ $(k=0,\cdots,m-1)$, then there exists an $\mathcal{E}'$-resolution lifting $\underline{\mathcal{D}}(C)$ of $\underline{\mathcal{D}}'(C')$ based on $(\pi,\mathcal{B})$ such that C subsumes C'; moreover, $\underline{\mathcal{D}}(C)$ is the embedding in $\mathcal{B}$ of a general $\mathcal{E}'$-resolution refutation $\underline{\mathcal{D}}_1(0)$ of $\{C_k: k<m\} \cup \mathcal{E}$, where $\mathcal{D}_1(0) \in \Delta'$.

Suppose that $\mathcal{D}'(C')$, π, $\mathcal{B}$, η, $\{C_k: k<m\}$, and $\underline{\mathcal{D}}''(0)$ are as above. Let π_1 be the branch mapping on $\underline{\mathcal{D}}''(0)$ such that $\pi_1(\mathcal{B}'_i) = C_k$ where $\pi(\mathcal{B}_i) = B_k$ and $\mathcal{B}_i$ is the branch of $\underline{\mathcal{D}}'(C')$ corresponding to the branch $\mathcal{B}'_i$ in $\mathcal{D}''(0)$. Let $\underline{\mathcal{D}}_1(0)$ be an $\mathcal{E}'$-resolution lifting of $\underline{\mathcal{D}}''(0)$ based on $(\pi_1,\{C_i: i<m\})$. Then $\underline{\mathcal{D}}_1(0)$ is in Δ' by strong liftability of Δ'. Let $\underline{\mathcal{D}}(C)$ be the embedding of $\underline{\mathcal{D}}_1(0)$ in $\mathcal{B}$. Then $\underline{\mathcal{D}}(C)$ has the desired properties provided that C subsumes $C' = (B'_0 - C'_0) \vee \cdots \vee (B'_{n-1} - C'_{n-1})$.

Clearly $C \leq (B_0 - C_0)\sigma_{\underline{\mathcal{D}}_1(0)} \vee \cdots \vee (B_{m-1} - C_{m-1})\sigma_{\underline{\mathcal{D}}_1(0)}$, by definition of $\underline{\mathcal{D}}(C)$ and $\mathcal{B}$. Define σ by

$$\sigma =_{df} [x\sigma_{\underline{\mathcal{D}}_1(0)}/x: \text{ x occurs in } B_k \text{ where } k<m]$$

Then it suffices to show that σ divides η, which is plausible

because $B_k\eta \leq B'_k \in \text{Base}(\underline{\mathcal{D}}'(C'))$ and $\sigma_{\underline{\mathcal{D}}'(C')} = \varepsilon$.

Each inference $(\mathcal{A}_i \vdash A_i)$ in $\underline{\mathcal{D}}_1(0)$ is the "lifting" of a corresponding inference $\mathcal{A}'_i \vdash A'_i)$ in $\underline{\mathcal{D}}''(0)$; moreover, the induced substitution θ_i of $(\mathcal{A}_i \vdash A_i)$ has the property that η_i divides η , where

$$\eta_i =_{df} [x\theta_i/x: \ x \text{ occurs in } \{C_k: k<m\}]$$

(If C_j is a premise in $\mathcal{A}_i$ then $C_j\eta_i \in \mathcal{L}_V$.) Moreover, if $\sigma_{\underline{\mathcal{D}}_1(0)} = \theta_0 \cdot \cdots \cdot \theta_{\ell-1}$ where $\{\theta_k: k<\ell\}$ is the set of substitutions induced by inferences in $\underline{\mathcal{D}}_1(0)$, then $\sigma = \eta_0 \cdot \cdots \cdot \eta_{\ell-1}$. It follows from these relations that σ divides η .∎

A refinement Δ is ground $\mathcal{E}$-complete provided that Δ is $\mathcal{E}$-complete on sets of constant clauses--i.e., if $\mathcal{C}$ is an $\mathcal{E}$-contradictory set of constant clauses then Δ contains a refutation of $\mathcal{C} \cup \mathcal{E} \cup [x=x]$.

Theorem 8. Suppose that Δ is a ground $\mathcal{E}$-complete $\mathcal{E}$-resolution refinement and Δ' is an $\mathcal{E}'$-complete generalized $\mathcal{E}'$-resolution refinement where $\mathcal{E} \supset \mathcal{E}'$. Then $\Delta \cdot \Delta'$ is a ground $\mathcal{E}$-complete generalized $\mathcal{E}'$-resolution refinement.

Proof. It suffices to show that if $(\{B'_i \vee q'_i: i<n\} \vdash C')$ is a ground $\mathcal{E}$-resolution inference where $C' = (B'_0 \vee \cdots \vee B'_{n-1})$ then Δ' contains a refutation $\underline{\mathcal{D}}'(0)$ of $\{q'_i: i<n\} \cup \mathcal{E}$, which is obvious because $\{q'_i: i<n\} \cup \mathcal{E}$ is inconsistent and Δ' is $\mathcal{E}'$-complete (hence complete).∎

From Theorems 7 and 8 we easily obtain the following:

Corollary 1. Suppose that Δ is a strongly liftable and ground $\mathcal{E}$-complete $\mathcal{E}$-resolution refinement and Δ' is a strongly liftable and ground $\mathcal{E}'$-complete $\mathcal{E}'$-resolution refinement where $\mathcal{E} \supseteq \mathcal{E}'$. Then $\Delta \cdot \Delta'$ is a strongly liftable and $\mathcal{E}$-complete $\mathcal{E}'$-resolution refinement.

Corollary 2. If $\mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ where $\mathcal{E}_n \cap \mathcal{L}_V \subseteq \mathcal{E}_0$ and $\mathcal{E}_k$ is a Horn system with renaming r_{s_k} $(k=0,\cdots,n)$ then $HR(\mathcal{E}_n,\succ_n,s_n) \cdot \cdots \cdot HR(\mathcal{E}_0,\succ_0,s_0)$ is a ground $\mathcal{E}_n$-complete $\mathcal{E}_0$-resolution refinement.

Let $\underline{\underline{\mathcal{E}}} = (\mathcal{E},\succ)$ where $\succ$ is an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E} \cap \mathcal{A}_V$. A refinement Δ is <u>strongly $\underline{\underline{\mathcal{E}}}$-complete</u> on a collection $\mathcal{U}$ of clause-sets provided that for each latent $\mathcal{E}$-contradiction $\mathcal{B}$ in $\mathcal{U}$ where $\mathcal{B}\theta$ is $\mathcal{E}$-contradictory and $\theta \in \Sigma_{\underline{\underline{\mathcal{E}}}}$, Δ contains a general refutation $\underline{\underline{\mathcal{D}}}$ of a finite set $\mathcal{C} \subseteq (\mathcal{B} \cup \mathcal{E} \cup [x=x])^{\sim}$ such that $\sigma_{\underline{\underline{\mathcal{D}}}}$ divides $\eta \cdot \theta$ for each invertible substitution η where $A\tilde{\eta} \in \mathcal{B}$ for each clause A in $\mathcal{C} - (\mathcal{E} \cup [x=x])^*$.

A clause-set $\{C_i : i < n\}$ is a <u>latent unit-clause set</u> provided that $\{C_i\theta : i < n\}$ is a unit-clause set for some substitution θ.

Theorem 9. Suppose that $\underline{\underline{\mathcal{E}}} = (\mathcal{E},\succ)$ where $\mathcal{E}$ is a set of equations and $\succ$ is an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E}$, and suppose that Δ_M is a strongly liftable ground $\mathcal{E}$-complete refinement and Δ_μ is a generalized $\mathcal{E}$-resolution refinement which is strongly $\underline{\underline{\mathcal{E}}}$-complete on latent unit-clause sets. Then $\Delta_M \cdot \Delta_\mu$ is $\mathcal{E}$-complete.

Corollary. Suppose that $\mathcal{E} \supseteq \mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ where $\mathcal{E} \cap \mathcal{L}_V \subseteq \mathcal{E}_0$ and $\mathcal{E}_k$ is a Horn system with renaming r_{s_k} $(k=0,\cdots,n)$, and suppose that Δ_μ is a basic refinement (consisting of normal deductions) which is strongly $(\mathcal{E}_0, \succ_0)$-complete on latent unit-clause sets, where $\succ_0$ is an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E}_0$. Then $HR(\mathcal{E}_n, \succ_n, s_n) \cdot \cdots \cdot HR(\mathcal{E}_0, \succ_0, s_0) \cdot \Delta_\mu$ is a normal $\mathcal{E}$-complete refinement.

Indication of proof. The refinement $\overline{\Delta} = (\overline{HR}(\mathcal{E}_n, \succ_n, s_n) \cdots \cdot \overline{HR}(\mathcal{E}_0, \succ_0, s_0)) \cdot \Delta_\mu$ is normal and $\mathcal{E}$-complete, by Theorem 7 (Corollary 2), Theorem 8 (Corollary 1), and Theorem 9. A corollary of Lemma 4 can be used to transform a member of $\overline{\Delta}$ into a member of $HR(\mathcal{E}_n, \succ_n, s_n) \cdot \cdots \cdot HR(\mathcal{E}_0, \succ_0, s_0) \cdot \Delta_\mu$.

Proof of Theorem 9. Let $\mathcal{B}$ be an $\mathcal{E}$-inconsistent clause-set. Then $\mathcal{B}^\sim$ includes a finite latent $\mathcal{E}$-contradiction $\mathcal{C}$ where $\mathcal{C}_\tau$ is an $\mathcal{E}$-contradiction and $\tau \in \Sigma_{\underline{\mathcal{E}}}$. Δ_M contains a refutation $\underline{\underline{\mathcal{D}}}'(0)$ of $\mathcal{C}_\tau \cup \mathcal{E}$, by ground $\mathcal{E}$-completeness. Suppose without loss of generality that π is a branch mapping for $(\underline{\underline{\mathcal{D}}}'(0), \mathcal{C} \cup \mathcal{E}^\sim)$. We "lift" $\underline{\underline{\mathcal{D}}}'(0)$ to a general refutation $\underline{\underline{\mathcal{D}}}''(0)$ in Δ_M in such a way that each inference in $\underline{\underline{\mathcal{D}}}''(0)$ can be realized by some embedding of a general refutation in Δ_μ. By "interpolating" these embeddings in $\underline{\underline{\mathcal{D}}}''(0)$ we obtain a refutation $\underline{\underline{\mathcal{D}}}(0)$ in $\Delta_M \cdot \Delta_\mu$.

Suppose that $(\{B_i' \vee q_i' : i < n\} \vdash C')$ is an inference in $\underline{\underline{\mathcal{D}}}''(0)$ with kernel $\{q_i' : i < n\}$, and that $\{B_i : i < n\}$ is a separated clause-set, consisting of conclusions of "lifted" subdeductions $\underline{\underline{\mathcal{D}}}_i(B_i)$ based on π and $\mathcal{D}''(B_i' \vee q_i')$, such that $B_i\theta \subseteq B_i' \vee q_i'$ where

$\theta \in \Sigma_{\underline{\mathcal{E}}}$ and B_i does not subsume $C' = (B'_0 \vee \cdots \vee B'_{n-1})$ $(i=0,\cdots,n-1)$. Let $C_i = \{q \in B_i : q\theta = q'_i\}$, so that $\{C_i : i<n\}$ is a latent $\mathcal{E}$-contradiction and a latent unit-clause set. It follows by strong $\underline{\mathcal{E}}$-completeness that Δ_μ contains a general refutation $\underline{\underline{\mathcal{D}}}'_j(0)$ of a finite set $\mathcal{C}_j \subseteq (\{C_i : i<n\} \cup \mathcal{E} \cup [x=x])^{\sim}$ such that $\sigma_{\underline{\underline{\mathcal{D}}}'_j(0)}$ divides $\eta\cdot\theta$ where η is an invertible substituion such that $A\eta \in \{C_i : i<n\}$ for each clause A in $\mathcal{C}_j - (\mathcal{E} \cup [x=x])^*$. Let $\underline{\underline{\mathcal{D}}}_j(C_j)$ be a general deduction obtained by embedding $\underline{\underline{\mathcal{D}}}'_j(0)$ in a separated set $\mathcal{B}_j \subseteq (\{B_i : i<n\} \cup \mathcal{E} \cup [x=x])^{\sim}$ and prefixing "variants" of the deductions $\underline{\underline{\mathcal{D}}}_i(B_i)$ $(i<n)$ where appropriate. Observe that $[\eta\cdot\theta/\sigma_{\underline{\underline{\mathcal{D}}}'_j(0)}] \in \Sigma_{\underline{\mathcal{E}}}$ and C_j subsumes C'. These observations complete the induction step in a systematic construction of $\underline{\underline{\mathcal{D}}}(0)$ from $\underline{\underline{\mathcal{D}}}'(0)$.

Theorem 9 and its corollary emphasize the crucial role of a generalized $\mathcal{E}$-resolution refinement Δ_μ which is strongly $\underline{\mathcal{E}}$-complete on latent unit-clause sets. Unfortunately, $ND(\mathcal{E},\succ)$ is not strongly $\underline{\mathcal{E}}$-complete on latent unit-clause sets; Theorem 3 asserts a weaker property, which yields the following strengthening of the Corollary to Theorem B (Abstract).

Theorem 10. Suppose that no non-unit clause of $\mathcal{E}$ contains a (positive) equation, $\mathcal{E} \supseteq \mathcal{E}_n \supset \cdots \supset \mathcal{E}_0$ where $\mathcal{E} \cap \mathcal{L}_V \subseteq \mathcal{E}_0$ and $\mathcal{E}_k$ is a Horn system with renaming r_{s_k} $(k=0,\cdots,n)$, $\succ'$ is an $\mathcal{E}'$-complexity ordering where $\mathcal{E}' \subseteq \mathcal{E}_0$, and $\Delta_M = HR(\mathcal{E}_n,\succ_n,s_n)\cdot\cdots\cdot HR(\mathcal{E}_0,\succ_0,s_0)$. Then $(\Delta_M \cdot ND(\mathcal{E}_0,\succ')) \cap ND(\mathcal{E}_0,\succ')$ contains a general refutation of $\mathcal{C} \cup \mathcal{E} \cup [x=x]$ for each $\mathcal{E}$-inconsistent clause-set

$\mathcal{C}$ wherein each non-unit clause contains at most one (positive) equation.

The proof is based on Lemma 12, which makes use of the set $\Sigma^-_{\underline{\underline{\mathcal{E}}}}$ of "constant" substitutions:

$$\Sigma^-_{\underline{\underline{\mathcal{E}}}} =_{df} \{\theta \in \Sigma_V : \text{if } x\theta \neq x \text{ then } x\theta \text{ is the first constant term in } [x\theta]_{\mathcal{E}} \text{ according to } \succ \ (x \in V_I)\}$$

Lemma 12. Let $\succ$ be a monotone partial ordering of V which well-orders constant terms, and suppose that $\{\theta_i : i \in N\}$ satisfies (i)-(iii):

(i) $\theta_k \in \Sigma^-_{\underline{\underline{\mathcal{E}}}_k}$ where $\mathcal{E}_k$ is a set of equations and $\underline{\underline{\mathcal{E}}}_k =_{df} (\mathcal{E}_k, \succ)$.

(ii) $\mathcal{E}_{k+1} = \mathcal{E}_k \cup \{[s_k = t_k]\}$.

(iii) $x\theta_{k+1} =_{\mathcal{E}_{k+1}} x\theta_k \quad (x \in V_I)$.

Then $\theta_{n+k} = \theta_n \quad (k \in N)$ for some $n \geq 0$.

Proof of Theorem 10. Suppose that $\mathcal{C}_0$ is an $\mathcal{E}$-inconsistent clause-set wherein each non-unit clause contains at most one equation. Suppose without loss of generality that $\mathcal{E}_0$ contains all of the equations

in $\mathcal{C}_0 \cup \mathcal{E} \cup [x=x]$.

It follows by ground $\mathcal{E}_0$-completeness of Δ_M that Δ_M contains a refutation $\underline{\underline{\mathcal{D}}}'(0)$ of some finite subset $\mathcal{C}_0'$ of $(\mathcal{C}_0 \cup \mathcal{E})^+$, where $\mathcal{C}_0' \leq (\mathcal{C}_0 \cup \mathcal{E})^{\sim} \theta_0$ for some θ_0 in $\Sigma^{-}\underline{\mathcal{E}}_0$. Let $\mathcal{E}_{\theta_0} = \mathcal{E}_0$, and let $\mathcal{E}_{\mathcal{C}_0}$ be the set of all equations which occur (positively) in clauses of $\mathcal{C}_0 \cup \mathcal{E}$.

Due to the assumptions that $\mathcal{E}_0$ contains all equations of $\mathcal{C}_0 \cup \mathcal{E} \cup [x=x]$ and that no non-unit clause of $\mathcal{C}_0 \cup \mathcal{E}$ contains more than one equation, each ($\mathcal{E}_0$-resolution) inference in $\underline{\underline{\mathcal{D}}}'(0)$ has the form

$$\mathcal{A}' \cup \{A' \vee p', B' \vee \tilde{q}'\} \vdash A' \vee B' \qquad (7)$$

where $\mathcal{A}'$ is a set of constant equations derived by "previous" inferences in $\underline{\underline{\mathcal{D}}}'(0)$ and $p' \models\!\!\!\models_{\mathcal{A}' \cup \mathcal{E}_0} q'$. Moreover, if p' is an equation, then $A' = 0$ because equations precede all other literals in each literal-ordering $\succ_i$ $(i=0,\cdots,n)$.

Case 1: $\underline{\underline{\mathcal{D}}}'(0)$ is $\underline{\mathcal{E}_0\text{-stable}}$, in the sense that $s =_{\mathcal{E}_0} t$ for eacn $\mathcal{E}_0$-resolution inference $(\mathcal{B}' \vdash [s=t])$ in $\underline{\underline{\mathcal{D}}}'(0)$. Then we define a general refutation $\underline{\underline{\mathcal{D}}}''(0)$ for $\mathcal{C}_0 \cup \mathcal{E}$ in $ND(\mathcal{E}_0, \succ)$ inductively as follows, assuming without loss of generality that $\mathcal{A}' = 0$ in (7).

Suppose (7) is an initial inference in $\underline{\underline{\mathcal{D}}}'(0)$. Then choose A, B in $(\mathcal{C}_0 \cup \mathcal{E})^{\sim}$ so that $A\eta \leq A' \vee p'$ and $B\eta \leq B' \vee \tilde{q}'$, selecting A and B according to some branch mapping for $\underline{\underline{\mathcal{D}}}'(0)$ so that all the initial premises of $\underline{\underline{\mathcal{D}}}'(0)$ will be separated.

Suppose that (7) is a non-initial inference of $\underline{\underline{\mathcal{D}}}'(0)$ and we have defined $\underline{\underline{\mathcal{D}}}''(A)$, $\underline{\underline{\mathcal{D}}}''(B),\eta$ so that $A\eta \leq A' \vee p'$ and $B\eta \leq B' \vee \tilde{q}'$,

but neither A nor B subsumes A' v B' . (Assume A,B separated.) Let θ be a m.g.s.u. of $\{\{p \in A\colon p\eta = p'\}, \{\tilde{q} \in B\colon \tilde{q}\eta = \tilde{q}'\}\}$. Let $A\theta \vee \underline{p\theta}$ be a simple factor of A on p where $p\eta = p'$, and let $B\theta \vee \underline{\tilde{q}\theta}$ be a simple factor of B on $\tilde{q}$ where $q\eta = q'$.

It follows by Theorem 3 (Corollary) that $ND(\mathcal{E}_0, \succ')$ contains a general realization $\underline{\underline{\mathfrak{D}}}_{A,B}$ for an $\mathcal{E}_0$-resolution inference

$$\{A,B\} \vdash (A\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}} - p\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}}) \vee (B\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}} - \tilde{q}\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}}) \tag{8}$$

where θ divides $\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}}$ and

$$[x\sigma_{\underline{\underline{\mathfrak{D}}}_{A,B}}/x\colon x \text{ occurs in } (A \vee B)] \text{ divides } \theta_0 .$$

Let $\underline{\underline{\mathfrak{D}}}"(C)$ be the result of prefixing $\underline{\underline{\mathfrak{D}}}"(A)$ and $\underline{\underline{\mathfrak{D}}}"(B)$ to $\underline{\underline{\mathfrak{D}}}_{A,B}$. It follows that $C\sigma_{\underline{\underline{\mathfrak{D}}}"(C)} \leq C'$. Let $x\eta = x\sigma_{\underline{\underline{\mathfrak{D}}}"(C)}$ for x in C .

Having defined $\underline{\underline{\mathfrak{D}}}"(0)$ in $ND(\mathcal{E}_0, \succ')$, we can easily transform $\underline{\underline{\mathfrak{D}}}"(0)$ into a refutation $\underline{\underline{\mathfrak{D}}}(0)$ in $(\Delta_M \cdot ND(\mathcal{E}_0, \succ')) \cap ND(\mathcal{E}_0, \succ')$. In the first place, if $\underline{\underline{\mathfrak{D}}}"(0)$ contains a simple factoring inference

$$B \vdash B\theta \vee \underline{\tilde{q}\theta} \tag{9}$$

where $\tilde{q}$ is r_{s_k}-negative and $\text{Level}_{\underline{\underline{\mathfrak{D}}}"(0)}(\tilde{q}) = k$ (§2.4.3), then the $\mathcal{E}_0$-resolution inference ($\{A,B\} \vdash C$) realization of which (9) is a constituent must be replaced by a chain of similar realizations wherein SF is not applied to B and the conclusion subsumes C . In other words, a realization of $\{A,B\} \vdash C$ containing (9) where $B = B_0 \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$ and θ is a m.g.u. of $\{q_0, \cdots, q_{n-1}\}$ is replaced by a subdeduction

containing n $\mathcal{E}_0$-resolution inference realizations. Assuming that $\underline{\mathcal{D}}''(0)$ has had all illegal SF-inferences removed in this manner, we obtain $\underline{\mathcal{D}}(0)$ in $(\Delta_M \cdot ND(\mathcal{E}_0,\succ')) \cap ND(\mathcal{E}_0,\succ')$ by reordering $\underline{\mathcal{E}}_0$-resolution inference realizations common to a hyper-$\underline{\mathcal{E}}_0$-resolution inference (realization) as in Lemma 4.

Case 2: $\underline{\mathcal{D}}'(0)$ is not $\mathcal{E}_0$-stable. Let $\underline{\mathcal{D}}'([s_0=t_0]\theta_0)$ be a subdeduction of $\underline{\mathcal{D}}(0)$ such that $s_0' \neq_{\mathcal{E}_0} t_0'$ and $\underline{\mathcal{D}}'(C')$ is $\mathcal{E}_0$-stable for each proper subdeduction of $\underline{\mathcal{D}}([s_0=t_0]\theta_0)$. Then by essentially the same argument as in Case 3 there exists in $(\Delta_M \cdot ND(\mathcal{E}_0,\succ')) \cap ND(\mathcal{E}_0,\succ')$ a deduction $\underline{\mathcal{D}}([s_0=t_0]\eta_0)$ (where $[s_0=t_0] \in \mathcal{E}_{\mathcal{C}_0}$) and η_0 divides θ_0. Observe that $s_0\eta_0 \neq_{\mathcal{E}_0} t_0\eta_0$ because $s_0\theta_0 \neq_{\mathcal{E}_0} t_0\theta_0$. Let θ_1 be the unique element of $\Sigma^-(\mathcal{E}_0 \cup [s_0=t_0]\eta_0,\succ')$ such that

$$x\theta_1 = \mathcal{E}_0 \cup [s_0=t_0]\eta_0 \; x\theta_0 \quad (x \in V_I)$$

and let $\mathcal{E}_{\theta_1} = \mathcal{E}_{\theta_0} \cup [s_0=t_0]\eta_0$.

It follows that $(\mathcal{C}_0 \cup \mathcal{E})^{\sim}\theta_1$ includes an $\mathcal{E}_{\theta_1}$-contradictory subset $\mathcal{C}_1' \subseteq (\mathcal{C}_0 \cup \mathcal{E})^+$, whence Δ_M contains a refutation $\underline{\mathcal{D}}_1'(0)$ of $\mathcal{C}_1'$.

Iterating the preceding case analysis with $\mathcal{C}_1'$ for $\mathcal{C}_0'$, θ_1 for θ_0, $\mathcal{E}_{\theta_1}$ for $\mathcal{E}_{\theta_0} = \mathcal{E}_0$, and $\underline{\mathcal{D}}_1'(0)$ for $\underline{\mathcal{D}}'(0)$, we find that "Case 1" is applicable at most a finite number of times, for otherwise Lemma 12 is contradicted. Thus, there exists a set $\mathcal{E}_{\theta_\ell} \subseteq \mathcal{E}^*_{\mathcal{C}_0}$, of equations derivable from $(\mathcal{C}_0 \cup \mathcal{E})^{\sim}$ by a deduction $\underline{\mathcal{D}}_0$ in $(\Delta_M \cdot ND(\mathcal{E}_0,\succ')) \cap ND(\mathcal{E}_0,\succ')$, such that $(\mathcal{C}_0 \cup \mathcal{E} \cup \mathcal{E}_{\theta_\ell})^+$ has an $\mathcal{E}_0$-resolution refutation $\underline{\mathcal{D}}_1'(0)$ which is $\mathcal{E}_{\theta_\ell}$-stable, whence

$\underline{\underline{\mathfrak{D}}}'_1(0)$ can be "lifted and interpolated" by a general deduction $\underline{\underline{\mathfrak{D}}}_1(0)$ in $(\Delta_M \cdot \mathrm{ND}(\mathcal{E}_0, \succ')) \cap \mathrm{ND}(\mathcal{E}_0, \succ')$ based on $(\mathcal{C}_0 \vee \mathcal{E} \cup \mathcal{E}_{\theta_\ell})^{\sim}$. Prefixing variants of subdeductions in $\underline{\underline{\mathfrak{D}}}_0$ where appropriate to support equations in $\mathcal{E}^{\sim}_{\theta_\ell} - \mathcal{E}^{\sim}_0$, we obtain a general refutation $\underline{\underline{\mathfrak{D}}}(0)$ of $(\mathcal{C}_0 \vee \mathcal{E})^{\sim}$ in $(\Delta_M \cdot \mathrm{ND}(\mathcal{E}_0, \succ')) \cap \mathrm{ND}(\mathcal{E}_0, \succ')$. ∎

4. RELATED RESEARCH

This chapter briefly reviews research which has contributed to or motivated the present investigation. The lack of comprehensiveness is partially compensated by the initial review of readily available overviews and textbooks.

The research most directly relevant to the technical results of this investigation is reviewed more carefully in §4.3. Relations to current research on calculi for enriched logical systems are described in §4.4.

Research on high-level programming languages with "deductive" capabilities, reviewed much more thoroughly in [9], is viewed briefly in §4.6 as a rapidly growing application area for specialized proof procedures.

4.1 Overviews and Textbooks

There are now several easily accessible overview articles and textbooks describing the research mentioned below and showing its relation to other areas of Artificial Intelligence (AI) research.

Nilsson's analysis of AI research [58] provides a lucid and comprehensive overview of research on proof procedures in the context of several types of deductive problem-solving systems: automatic theorem proving, automatic programming, question answering, and robotic problem-solving systems. This survey clearly documents a growing trend toward recognition of the importance of secondary "heuristic advice" information along with the primary (descriptive or axiomatic) information in the data bases of deductive problem-solving systems.

Resolution-based calculi and their applications in AI are thoroughly treated in two recent textbooks [54,12]. Luckham [47] provides a concise development of the basic results supporting the resolution principle of J. A. Robinson (essentially a combination of the Cut and Factoring rules (§2.2.2).

In [67], Robinson presents a more general version of his resolution principle which admits resolution inferences (§2.1.1) but not $\mathcal{E}$-resolution inferences in general. This article includes a useful overview of research in mechanical theorem proving, augmented by a comprehensive bibliography.

4.2 Formal Foundations and Background

Completeness and undecidability. Well known studies in the foundations of mathematics have shown that the logical consequence relation $\models$ is semi-decidable but not decidable ; similar conclusions hold for $\models_{\mathcal{E}}$ under the assumption (implicit throughout this report) that $\mathcal{E}$ is consistent. Gödel (1930) and others presented effective calculi for first-order logic and showed them to be complete.

Resolution principle. Following the completeness results for first-order predicate calculi and the later advent of high-speed, programmable digital computers, a number of investigators began to experiment with basically complete proof-procedures for first-order predicate logic [66,17]. The resolution principle formulated by J. A. Robinson [63] grew directly out of efforts to systematically eliminate major and (practically speaking) combinatorially disastrous

sources of inefficiency in these procedures. This principle, embodied in the simple Factoring and Cut rules of inference (§2.2.2) is based on the concept of most general (simultaneous) unifier developed by Prawitz [59] and refined by Robinson [67].

Heuristic methods. The latter part of [63] is devoted to several uniform (problem-independent) heuristics for eliminating redundant clauses and inferences. Heuristics (both uniform and problem-dependent) for the control of refutation procedures based on the resolution principle have been the subject of intensive investigations for nearly a decade. Several refinements of resolution, while originally conceived of as being heuristic in the sense of being "the sort of thing an intelligent human problem-solver would do" [63,p. 118], were subsequently shown to preserve the refutation completeness property of resolution. These developments are reviewed more extensively in Robinson's article [66].

Refinements. In [48], D. Luckham defined a general class of restrictions on binary (or "pairwise") resolution, which he called refinements, deriving completeness and compatability results for model partition and ancestry filtering refinements. The latter, also known as linear resolution [4,43,46] was probably the first example of a refinement which cannot be defined without reference to the proof-trees supporting the premises of admissible inferences.

Model partition, or resolution relative to a model, is one of several generalizations of P_1-deduction or its "non-basic" form hyper-resolution, defined and shown to be complete by J. A. Robinson in 1965 [64]. Slagle [73] and Kowalski [40] investigated certain literal-

orderings and renamings (§2.1.2) in conjunction with these refinements.

Refinements for equality calculi were obtained from each of the above types of refinements by imposing restrictions on the use of paramodulation [42,78]. Hyper-E-resolution [3] was an extension of hyper-resolution to equality theory, and renamable paramodulation [12] incorporated renamings and certain literal-orderings into this refinement. Kowalski [41] notes the equivalence of refinement-restricted uses of equality-axioms and special inference rules such as paramodulation.

Special inference rules for fragments of set theory and other theories of partial order are investigated by Slagle [75,76]. Completeness and efficiency results for these rules are based on previous results for refinements and axiomatizations of set membership and partial orders.

Special refinements and search strategies for Horn systems (§3.1.1) are investigated by D. Kuehner in [45], where he shows the completeness of a purely linear (or input) resolution strategy for Horn systems. Since the addition of equality axioms to a Horn system yields a Horn system, it seems likely that these results can be extended to first-order logic with equality.

Functional reflexivity axioms $[fx_1 \cdots x_n = fx_1 \cdots x_n]$ are required to be present for each of the above completeness results pertaining to resolution augmented by paramodulation. L. Wos and G. A. Robinson discuss possible ways of showing that only simple reflexivity ([x=x]) is needed for completeness; however, their conjecture to this

effect [64] remains open.[1]

Performance analyses, particularly by empirical measures of efficiency ratio (§B.2) have been conducted by several of the above authors (e.g., [51] and [63]). The foundations for a theory of efficiency for proof procedures are investigated by R. Kowalski in his doctoral thesis and discussed by B. Meltzer in [51]. The development in §B is based on essentially the same model of proof procedures used by Kowalski.

4.3 Research on Equational Simplification Refinements

4.3.1 Studies in Combinatory Logic

Much of the research described below is motivated by the word problem for an equational system $\mathcal{E}$:

Given terms u and v, decide whether or not $u =_{\mathcal{E}} v$. (1)

It is well known that the word problem is unsolvable for many finite equational extensions of Semigroup or Group Theory [71].

However, basic research by Church [14], Rosser, and Curry [16] on lambda-calculi and their equational equivalents, the combinatory calculi [30], have established a paradigm for solutions to the word problem: they investigated several finite equational reduction systems $(\mathcal{E}, \rightarrow^*)$ (as defined in §2.3.4) and showed them to be complete (§2.3.4):

---

[1] Unit-completeness results for paramodulation with simple reflexivity have been derived independently by Richter [60] , and myself (§3.2.1, Theorem 3). It appears quite likely that partial completeness results for the non-unit case (e.g., Theorem B) will be extended in the near future.

If $r =_{\mathcal{E}} s$ then $r \to^* t$ and $s \to^* t$ for some term t. The various completeness theorems became known as Church-Rosser theorems.

The following basic combinatory system (IKS) has been investigated extensively. V_R is empty, and $V_F = V_F^0 \cup V_F^2$ where $V_F^0 = \{I,K,S\}$ and $V_F^2 = (\_\ \_)$; $(\_\ \_)$ is the binary application operator and I,K,S are the basic combinators. Defining $(u_1 \cdots u_n)$ for $n > 2$ by

$$(u_1 \cdots u_n) =_{df} ((u_1 u_2) u_3 \cdots u_n) ,$$

we define $(IKS) = \{(I),(K),(S)\}$:

I: $[(Ix) = x]$

K: $[(Kxy) = x]$

S: $[(Sxyz) = (xz(yz))]$

It turns out that every partial recursive function is represented in $=_{(IKS)}$ by some constant term, in a very natural sense [30].

A basic Church-Rosser theorem asserts that $((IKS), \to^*_{(IKS)})$ is complete. However, the word problem for (IKS) is undecidable, and $NF((IKS), \to^*_{(IKS)})$ is not an (IKS)-normal form.

4.3.2 Completeness of Reduction Systems Based on Complexity Orderings

Knuth and Bendix [39] derive a completeness characterization and a procedure for computing complete reduction systems, working with finite, total, equational reduction systems $(\mathcal{E}, \succ_w)$ where $\succ_w$ is defined from a weighting function w as in §2.3.5. Their basic result is an effective characterization of complete (and therefore canonical) reduction systems in this class (Proposition 2 in §C.4).

They describe a partial algorithm (essentially the one specified in §3.2.3 for deriving a complete system $(\mathcal{E}', \succ_W)$ from $(\mathcal{E}, \succ_W)$, and they illustrate its applications to the word problem by means of numerous computer-generated "completions" of simple equational systems (e.g., the Group Theory example in §C.2).

The completeness results for basic refinements of the form $ND(\mathcal{E}, \succ)$ in §3.2 were motivated by, and constitute an extension of, the research of these two investigators.

4.3.3 Refinements Based on Composition with $\mathcal{E}$-Canonical Mappings

Suppose that $\mathcal{E}$ is an equational system and ν is an $\mathcal{E}$-canonical mapping on $\mathcal{T}_V \cup \mathcal{C}_V$, whence $(s =_{\mathcal{E}} t)$ iff $(\nu(s) = \nu(t))$. Given a calculus Γ over $\mathcal{C}_V$, it is natural to ask how ν may be used to "refine" Γ for greater efficiency in proof procedures.

Plotkin [57] describes his objectives as finding a complete calculus Γ such that the calculus Γ_ν, defined by

$$(\mathcal{B} \vdash_{\Gamma_\nu} C') =_{df} (\mathcal{B} \vdash_{\Gamma} C \text{ where } \nu(C) = C') \qquad (1)$$

is also complete[1]. Actually, there is a straightforward modification of $ND(\mathcal{E}, \rightarrow^*_{\mathcal{E}})$ which appears to satisfy this objective, provided that $u \rightarrow^*_{\mathcal{E}} \nu(u) \in NF(\mathcal{E}, \rightarrow^*_{\mathcal{E}})$ $(u \in \mathcal{T}_V \cup \mathcal{C}_V)$. However, Plotkin's development

---

[1] Plotkin uses N for ν, and states his objective thus: "We shall look for a complete set of rules $r_1, \cdots, r_m$ such that $r_1 \circ N, \cdots, r \circ N$ is also a complete set. (If r is a rule, $r \circ N$ is the rule which outputs $N(C)$ iff r outputs, with the same inputs, C)". [57, p. 74]. (Underlining mine).

is quite different; parts of it are outlined in the following paragraphs, given $\mathcal{E}$ and ν as above.

Instantiation[2] and composition are defined as follows:

$$u * \sigma =_{df} \nu(u\sigma) \qquad (u \in \mathcal{T}_V \cup \mathcal{C}_V);$$

$$\eta * \theta =_{df} \nu(\eta\theta) \ ;$$

where $\nu(\sigma) =_{df} [\nu(x\sigma)/x: x\sigma \neq x]$.

A substitution σ $\mathcal{E}$-unifies a set U provided that $u * \sigma = v * \sigma$ for all u,v in U.

Example 1.[3] Let $\mathcal{E} = \{[(x \cdot y) \cdot z = x \cdot (y \cdot z)]\}$, and let $\nu(s)$ be the unique term in $[s]_{\mathcal{E}} \cap NF(\mathcal{E}, \rightarrow_{\mathcal{E}}^{*})$ (whence $\nu((w \cdot x) \cdot (y \cdot z)) = (w \cdot (x \cdot (y \cdot z)))$). Let $U = \{w \cdot x, a \cdot (b \cdot c)\}$. $[a \cdot b/w, c/x]$ and $[a/w, (b \cdot c)/x]$ are both $\mathcal{E}$-unifiers of U.

Definition.[4] σ is a simplest $\mathcal{E}$-unifier of U provided that

(i) σ $\mathcal{E}$-unifies U;

(ii) If $\eta * \theta = \nu(\sigma)$ and η $\mathcal{E}$-unifies U then θ is $\mathcal{E}$-invertible: $\theta * \theta' = \varepsilon$ for some θ'.

Definition.[4] η $\mathcal{E}$-divides σ provided that $\eta * \theta = \nu(\sigma)$ for some θ.

Observe that in Example 1, $[a \cdot b/w, c/x]$ and $[a/w, (b \cdot c)/x]$ are both simplest $\mathcal{E}$-unifiers of U, and that neither $\mathcal{E}$-divides the other.

---

[2] Plotkin refers to this operation on $\mathcal{T}_V \times \Sigma_V$ as ν-application.

[3] All examples taken from [59].

[4] My own, to simplify exposition without altering content. (I am assuming only that $\eta * \theta = \nu(\eta) * \nu(\theta)$ in this presentation.)

Thus, no "$\mathcal{E}$-unification algorithm" based on * could possibly return a single "simplest $\mathcal{E}$-unifier" for each finite $\mathcal{E}$-unifiable set.

Definition. σ and σ' are $\mathcal{E}$-independent provided that neither σ $\mathcal{E}$-divides σ' nor σ' $\mathcal{E}$-divides σ.

Example 2. Let $\mathcal{E},\nu$ be as in Example 1. The set of simplest $\mathcal{E}$-unifiers of $\{g(x,x\cdot a), g(y,a\cdot y)\}$ contains the infinite set of mutually $\mathcal{E}$-independent substitutions $\{[a^n/x,a^n/y]: n \in N\}$.

Observing that with problems such as these one must invent a special unification algorithm for each equational system (p. 74), Plotkin defines the function of an "$\mathcal{E}$-unification" procedure as the generation from U of a set containing exactly one representative of each "$\mathcal{E}$-variant" class of simplest $\mathcal{E}$-unifiers of U.

Finally Plotkin defines two inference rules for use with $\mathcal{E},\nu$: one corresponding to (factoring and resolution), the other corresponding to (factoring and paramodulation). In the former case, inferences of the form

$$\{A \vee C, B \vee \tilde{D}\} \vdash (A \vee B)\cdot\sigma \qquad (2)$$

where

(i) $A \vee C$ shares no variables with $B \vee \tilde{D}$;

(ii) $C \neq 0, \tilde{D} \neq 0$, and neither C nor $\tilde{D}$ contains an equation; and

(iii) σ is a simplest $\mathcal{E}$-unifier of $C \cup D$, where $\tilde{D} =_{df} \{\tilde{q}: q \in D\}$

are admitted. In other words, given $C,\tilde{D}$ where $C \cup D$ is $\mathcal{E}$-unifiable, we generate each inference (2) where σ is a simplest substitution such that $\nu(C\sigma) \cup \nu(D\sigma)$ is a singleton.

Remark. In view of the stated objective (1), we might expect resolution inferences of the form

$$\{A \vee C,\ B \vee D\} \vdash (A \vee B) * \sigma \qquad (3)$$

where σ is a m.g.u. of $C \cup D$ to be defined. Such inferences do not require the computation of simplest $\mathcal{E}$-unifiers. Plotkin does not consider (3), even though it corresponds to his stated objective and (2) does not. (3) is essentially the form of factoring-resolution incorporated into $ND(\mathcal{E}, \rightarrow^*)$.

In any case, Plotkin proves refutation completeness of his calculus on functionally reflexive $\mathcal{E}$-inconsistent sets of clauses, assuming the existence of a complete generator for simplest $\mathcal{E}$-unifiers. He argues that this calculus should yield greater efficiencies than the resolution-paramodulation calculus which uses ordinary unification and gives no special treatment to equations of $\mathcal{E}$.

4.3.4 Demodulation and Simplification Strategies

The idea of using the current set of derived equations as the basis for a "simplification mapping" to be applied to other derived clauses has been investigated extensively by L. Wos et al. [81], and by Luckham et al. [1] on the Interactive Theorem Prover.

"Demodulation" with respect to a set $\mathcal{E}$ of equations is defined in [81] from the $\mathcal{E}$-reducibility relation $\rightarrow\!\!>_{\mathcal{E}}^*$ defined on terms by

$(u[r] \rightarrow\!\!>_{\mathcal{E}} u[t\theta]) =_{df}$ $\mathcal{E}$ contains $[s=t]$ where θ is a m.g.u. of $\{r,s\}$, $r=s\theta$, and $t\theta$ contains strictly fewer symbols than $s\theta$.

($\twoheadrightarrow^*_{\mathcal{E}}$ is the reflexive, transitive extension of $\twoheadrightarrow_{\mathcal{E}}$, §2.3.2.) Evidently, $NF(\mathcal{E}, \twoheadrightarrow^*_{\mathcal{E}})$ is an $\mathcal{E}$-normal form for $\mathcal{J}_V \cup \mathcal{C}_V$. As the authors note, it is not in general an $\mathcal{E}$-canonical form.

Demodulation of a derived clause A consists of replacing A with C where $A \twoheadrightarrow^*_{\mathcal{E}} C \in NF(\mathcal{E}, \twoheadrightarrow^*_{\mathcal{E}})$.

Noting the incompleteness of refinements wherein exhaustive demodulation is used, Wos et al. define a finitary variant of binary resolution, called k-modulation, wherein some of the demodulation replacements can be "undone" before the resolvents of two premises are generated.

A similar simplification strategy, based on a user-specified list of equations, is a built-in strategy of the Interactive Theorem Prover. The conclusion of each inference is "simplified" (e.g., demodulated) with respect to this list before being added to the proof-procedure's current deduction.

The utility of these strategies is amply documented in [81] and in [52]. The data of Wos et al. show that appropriate strategies based on demodulation and k-modulation can significantly reduce the cost of refuting typical inconsistent clause-sets corresponding to theorems such as "Boolean rings of characteristic 2 are commutative." Cost (in time) for PG5 with demodulation was typically one-tenth the cost for a similar program (PG4) without demodulation [81, Table 1].

4.4 Enriched Logics and Their Calculi

First-order logics and their proof procedures are well suited for decision-making within states or world-models, but poorly suited for reasoning about actions or processes based on transitions between states

or world models. Temporal and other modal logics [11,44,50] are needed for robot design and automated programming applications, which can afford neither the inefficiency of a uniform embedding of space-time in a predicative framework [23] nor the conceptual chaos likely to ensue from an ad hoc synthesis of deduction and simulation. The similarity between certain modal logic systems [44] and the formal structure of experimental problem-solving systems such as STRIPS [20] suggests the possible emergence of an appropriate interface, in which state-transformation operators are selected by deductively evaluating their applicability conditions in current and alternative world models. Such an interface would have the desirable property of making the dynamic or modal aspects of the environment more or less transparent to the "classical" deductive component of the system.

Independently of the modal dimension, there is a demand for increased richness in the predicate logics amenable to automated proof procedures. Both mathematicians and users of higher-level programming languages find it natural to work with type-structured languages instead of pure first-order languages with a single type of individual variable. The mathematician using a deductive problem-solving system will frequently be communicating and thinking in terms of a higher-order logic, quantifying and defining predicates over infinite sets of functions and relations as well as over individuals. Higher-level programming languages have equally sophisticated type structures, allowing many individual types (sorts) as well as functional types, denoting classes of functions which take individuals and other functions as arguments.

Extensions of resolution to higher-order logic have been discussed [24], advocated [22] or proposed [70,5] by a number of investigators, although no really satisfactory extensions have been described. Andrews' proposed extension, modeled on Henkin's theory of simple types [27], abandons the "most-general inference" property of first-order resolution systems, relying instead on a "truth-functional" cut rule and a separate instantiation rule.

The doctoral dissertation of Gould [21] has occasionally been referred to as evidence for the non-existence or, at best, extreme complexity, of a unification algorithm for a calculus with abstraction ("λ-expressions") and conversion rules. The dissertation itself, however, is a brief, technically correct demonstration that Gould's particular concept of unification or matching, in λ-calculus with types, is in fact a computationally untenable synthesis of unification (or instantiation) and simplification (λ-conversion). Nothing is said of the (dubious) justification for Gould's definition of "matching" in higher-order logic as an analogue of the first-order unification algorithm, nor is the possibility of a more feasible approach discussed.

In essence, Gould's approach is the same as the approach investigated by Plotkin (§4.3.3) for incorporating "algebraic simplification" or "evaluation" (relative to an equational system $\mathcal{E}$) into the unification process. In Gould's case, $\mathcal{E}$ corresponds to a set of "typed" versions of equations in (IKS) (§4.3.1).

In view of the alternative approach noted in §4.3.3, it appears that the "difficulties" encountered in extending resolution proof procedures to higher-order logics [33,51] are purely formal difficulties

associated with one approach to the desired extension; these difficulties do not appear to be intrinsic to the extension itself.

4.5 Theorem-Proving Systems

The Interactive Theorem Prover of Allen and Luckham [1] is basically a complete refutation procedure augmented by a flexible man-machine interface incorporating powerful facilities for the specification of refinements and search strategies. The system has been used to obtain proofs of new mathematical results announced without proof in Notices of the American Mathematical Society.

A much more specialized proof procedure for Integer Arithmetic was developed by Floyd and King [37] as part of a computer program verification system. Incorporating powerful facilities for algebraic simplification and linear system solving on its intended domain, this refutation procedure efficiently decides a small but useful class of propositions[1] and quickly gives up on the others.

The on-line systems of Bledsoe et al. [7,8] are being used to explain the potential of specialized theorem-proving systems in domains such as real analysis, set theory, and topology. Based on powerful sub-goaling facilities, these systems incorporate limited facilities for handling typed variables (e.g., interval types in real analysis). A linear system solver makes good use of these facilities.

The systems of Luckham and Bledsoe reflect and facilitate the important role of the human operator in many deductive problem-solving

---

[1] Averaging about ten seconds each on an IBM 360 Model 67 [37].

applications. Appreciation for this role in theorem-proving applications is reflected in the following remarks by Bledsoe [6]:

> There is a real difference between doing some mathematics and being a mathematician. The difference is principally one of judgement: in the selection of a problem (theorem to be proved); in determining its relevance; in choosing reference theorems to help prove the given theorem; in selecting techniques for use in the proof; in knowing when to abandon one line of attack in favor of another, perhaps using as evidence information derived from the attempted proof; in knowing when to prove a convenient lemma; in knowing how to restructure the proof into a more lucid form once a proof has been found; in knowing how to balance the search for a proof with the search for a counterexample. It is precisely in these areas that machine provers have been so lacking. This kind of judgement has to be supplied by the user, and hence the system is in reality a man-machine system.

4.6 Deductive Problem-Solving Languages and Systems

One of the primary functions of Computer Science is to develop formal languages which facilitate computer usage in various applications areas. Language in this sense refers to syntax, semantics (as defined by interpreters and compilers), and various support facilities (e.g., on-line editing and date-base management facilities). Research in AI has stimulated the development of several new languages designed to facilitate construction of the complex deductive problem-solving systems needed for automatic theorem proving, automatic programming, and robot problem-solving applications.

There have been two distinct trends in language development. One, the "structured programming" trend, is characterized by a conservative approach toward new built-in concepts and a strong emphasis on systematic program verification methods. Another, the "ad hoc programming" trend, is characterized by liberal experimentation with built-in concepts designed for "high-level" programming, where the level of a programming language reflects "the extent to which a programmer may specify what he wants accomplished without specifying how it is to be done..." [9].

The "structured programming" trend is represented by languages such as Pascal [31] and Concurrent Pascal [10]. Axiomatic formalizations of semantics and program verification systems are currently being developed for Pascal [31] .

The "ad hoc programming" trend is represented by languages such as Planner [29] , Conniver [9], and QA4 [9]. Unlike the "structured programming" languages, these languages incorporate "deductive" facilities directly in the form of subproof generators, search strategies, and pattern-directed procedure invocation [9].

The terms "structured" and "ad hoc" are not intended to indicate any intrinsic distinction or dichotomy between the two classes of programming languages references above. Indeed, I believe that the basis for any such distinction will disappear within a few years. The "ad hoc programmers" developed their philosophy of language design at a time when no convincing method for systematic or hierarchical construction of specialized problem-solvers was available. The "structured programmers" are just beginning to develop promising approaches to

automatic program generation and verification [86], and it will be a while before these methods are fully integrated with languages such as Conniver and QA4. The synthesis of "structured" and "ad hoc" programming concepts will probably result in very high-level programming languages with built-in facilities for program generation and verification, based partially on powerful proof-procedures whose refinements and search strategies will be developed along with the application-system being programmed.

5. CONCLUSIONS

This chapter summarizes accomplishments (§5.1) and limitations (§5.2) of the research described in Chapters 1-3 and appendices. Extensions and applications are outlined in §5.3.

The principal limitations discussed in §5.2 are the lack of empirical performance measures for the normal refinements developed in this research and the limited domain of completeness for refinements of the form $\Delta_M \cdot ND(\mathcal{E}, \succ)$. Ways of overcoming these limitations are discussed in §5.3.1 and §5.3.2.

Extensions of normal refinements and proof procedures to enriched logics (§4.4) are outlined in §5.3.4 and §5.3.5. While these sections are concerned more with the definition of the enriched logic than with the details of the extensions, the extensions do appear to be feasible.

Extensions to type-structured logics based on the lambda-calculus syntax of Church are described in §5.3.4. The unification algorithm for such extensions is a natural restriction of the first-order unification algorithm based on a compatability requirement for types of variables and terms. Maintenance of terms in normal form is ensured by the use of an extension of $ND(\mathcal{E}, \succ)$ in the proof procedures.

A class of type-structured "procedural logics" is outlined in §5.3.5. These logics and their refutation procedures, if successfully developed on the basis of normal refinements, will be well suited for reasoning about (or reasoning within) the computational procedures of deductive problem-solving systems.

5.1 Accomplishments

5.1.1 Structured Design of Specialized Refinements

A basic contribution of this research has been the introduction of structured programming methodology into the design and specification of specialized proof procedures. The procedure Ref (§1.3.6) exemplifies the separation of structural knowledge (Δ) and procedural knowledge (Enq) in the specification o proof procedures.

The concepts of Γ-closure completeness for refinements (§1.3.4) and fairness for search strategies (§1.3.5) formalize useful guidelines for the design of proof procedures, as shown by Propositions 5 and 6 in §1.3.

However, the feasibility of the structured programming methodology in this context depends upon the availability of a rich class of generalized $\mathcal{E}$-resolution refinements and a method of composing them which preserves their completeness and efficiency properties. The principal contribution of this research has been the definition and analysis of a class of such refinements.

The concept of a generalized $\mathcal{E}$-resolution inference relativizes the generalized resolution principle of Robinson [67] to an arbitrary relative consequence relation $\models_{\mathcal{E}}$, enabling the proof-procedure designer to partition the design problem into two logically independent stages: the design of a (generalized) $\mathcal{E}$-resolution refinement (and corresponding search strategy), and the design of a refinement for the effective and efficient realization of (generalized) $\mathcal{E}$-resolution

inferences by lower-level inferences.

The normal composition operation (§2.4.1) provides a natural method for composing a generalized $\mathcal{E}$-resolution refinement with a generalized $\mathcal{E}'$-resolution refinement where $\mathcal{E} \supseteq \mathcal{E}'$, so as to obtain a generalized $\mathcal{E}'$-resolution refinement which preserves useful features of its constituents (§3.3). It is this preservation property which makes it feasible for the designer to reduce the (typically intractable) problem of designing a refinement for $\mathcal{E}$ by designing and then composing refinements for well understood subsets of $\mathcal{E}$.

5.1.2 Completeness Results for Hyper-$\mathcal{E}$-Resolution

Hyper-$\mathcal{E}$-resolution and the "equivalent" $\mathcal{E}$-resolution refinement HR($\mathcal{E}$,$\succ$) provide a specific class of (generalized) $\mathcal{E}$-resolution refinements with known completeness properties. These refinements are useful in the design of normal refinements, provided that $\mathcal{E}$ is a Horn system with the renaming specified by $\underline{\underline{\mathcal{E}}}$.

The completeness result for HR($\mathcal{E}$,$\succ$,s) (Theorem 1) illustrates several useful concepts in the design and analysis of $\mathcal{E}$-resolution refinements: independence of completeness and effectiveness, ground completeness and the excess literal parameter (Proposition S in §3.1.2), and strong liftability.

5.1.3 Design of Resolution Micro-Refinements

The concept of a normal refinement and the associated convention (§2.4.2) focuses attention on two major stages in the design of proof procedures: the non-unit stage is concerned with the

specification of a refinement for $\mathcal{E}$ in terms of an $\mathcal{E}_0$-resolution refinement Δ_M where $\mathcal{E}_0 \cap \mathcal{L}_V \subseteq \mathcal{E}_0 \subseteq \mathcal{L}_V$; the unit stage (Δ_μ) is concerned with the efficient realization of $\mathcal{E}_0$-resolution inferences by means of normal basic deductions.

Theorem 9 describes a property of Δ_μ (strong $\underline{\mathcal{E}}_0$-completeness on latent unit clause sets) which yields a general completeness result under composition with a wide variety of $\mathcal{E}_0$-resolution refinements.

The resolution micro-refinement ND($\mathcal{E}_0$,$\succ$), while not strongly $\underline{\mathcal{E}}_0$-complete, does have a more restrictive completeness property (Theorem 3) which yields a less general (but nevertheless useful) completeness result for certain normal refinements (Theorem 10).

ND($\mathcal{E}_0$,$\succ$) eliminates many redundant and irrelevant inferences from unit refutations, as evidenced by the examples in §1.1.3 and §D.4. Theorems 9 and 10 together strongly suggest the desirability of extending ND($\mathcal{E}_0$,$\succ'$) to a refinement Δ_μ which behaves like ND($\mathcal{E}_0$,$\succ$) (on the known domain of completeness of ND($\mathcal{E}_0$,$\succ'$)) and is also strongly $\mathcal{E}_0$-complete on latent unit clause sets. Thus, these limited results and observations provide useful guidelines for subsequent research on specialized refinements which preserve general completeness.

5.2 Limitations

Empirical Support. The concept of an application environment ($\mathcal{U}_\mathcal{E}$,μ) provides a precise and realistic formal framework for performance evaluation and comparison of specialized proof procedures and,

indirectly, the refinements upon which they are based (§B). However, this framework has yet to be applied to the performance-evaluation of normal refinements (§5.3.1). The examples and performance estimates in §1.1.3, §C, and §D are at best indications that the normal refinements and completeness results developed in this research may constitute a significant contribution to the design of specialized proof procedures; such claims rely heavily upon the reader's experience and intuitions in the absence of a more thorough empirical investigation.

Analytical support (other than completeness results) for the alleged usefulness of this research is also minimal. Many potential users of normal proof procedures would be more convinced by empirical "efficiency" results than by hypothetical-analytical arguments such as the following.

Example. Suppose that Δ_M is an $\mathcal{E}$-resolution refinement and Δ_μ, Δ'_μ are $\mathcal{E}$-resolution micro-refinements. Suppose that the expected "complexity" of a refutation of a member of $\mathcal{U}_\mathcal{E}$ in Δ_M is n $\mathcal{E}$-resolution inferences. $(\mathcal{U}_\mathcal{E}, \mu)$ and Δ_M define an application environment $(\mathcal{U}_M, \mu_M)$ which reflects the class and distribution of refutable sets which will be encountered by Δ_μ (or Δ'_μ) in the context of $\Delta_M \cdot \Delta_\mu$ (or $\Delta_M \cdot \Delta'$). Suppose that the expected number of alternative refutations of a member of $\mathcal{U}_M$ in Δ_μ is m, and that the expected number of alternative refutations of a member of $\mathcal{U}_M$ in Δ'_μ is m/2. Then, assuming that a "breadth first" search strategy is used, we should expect the expected efficiency ratio of a procedure based on $\Delta_M \cdot \Delta_\mu$ to be close to $r'/2^n$ where r' is the expected efficiency ratio of a similar procedure based on $\Delta_M \cdot \Delta'_\mu$, because a

refutation in Δ_M with n inferences has (approximately) m alternative realizations in $\Delta_M \cdot \Delta'$ and $(m/2)^n$ alternative realizations in $\Delta_M \ \Delta'_\mu$.

Domain of completeness. I doubt that general completeness results for normal refinements of the form $\Delta_M \cdot ND(\mathcal{E}_0, \succ')$ can be extended significantly beyond the domain of clause-sets wherein each clause contains at most one equation (Theorem 10), even though I have no specific example to support this doubt. The following intuitive argument may suggest a specific example and a subsequent modification of $ND(\mathcal{E}, \succ)$ which does have the strong $\underline{\underline{\mathcal{E}}}$-completeness property required for the proof of Theorem 9.

Consider a refinement $\Delta = \Delta_M \cdot ND(\mathcal{E}, \succ)$ where Δ_M is a strongly liftable ground $\mathcal{E}$-complete $\mathcal{E}$-resolution refinement, $\mathcal{E}$ is a set of equations, and $\succ$ is an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E}$. In order to prove general $\mathcal{E}$-completeness of Δ by the argument used for Theorem 10, it suffices to verify the following:

Proposition. Suppose that θ is in $\Sigma^-_{\underline{\underline{\mathcal{E}}}}$, that $x\theta \neq x$ for each variable x occurring in $\{B_i \vee q_i : i < n\}$, and that

$$\{(B_i \vee q_i)\theta : i < n\} \vdash C'$$

is an $\mathcal{E}$-resolution inference where $C' = (B_0\theta - q_0\theta) \vee \cdots \vee (B_{n-1}\theta - q_{n-1}\theta)$. Let $C = \{q \in B_i \vee q_i : q\theta = q_i\}$ $(i=0,\cdots,n-1)$. Then $ND(\mathcal{E}, \succ)$ contains a general refutation $\underline{\underline{\mathcal{D}}}(0)$ of $\{C_i : i < n\} \cup \mathcal{E}$ such that for each invertible substitution η where $C\eta \in \{C_i : i < n\}$ $(C \in \text{Base}(\underline{\underline{\mathcal{D}}}(0)) - \mathcal{E}^*)$, $\sigma_{\underline{\underline{\mathcal{D}}}(0)}$ divides $\eta \cdot \theta$.

Theorem 3 tells us that this proposition holds when $\models_{\mathcal{E}} q_i$ for each equation q_i in $\{q_i : i < n\}$. However, if q_j is an equation and $B_j - C_j \neq 0$ then we cannot assume that $q_j \in \mathcal{E}^*$ because

to do so would require us to use q_j (thereby introducing clauses of B_j- C_j) in subsequent $\mathcal{E}$-resolution inference realizations. (This remark should be comprehensible in the context of the proof of Theorem 10.) The above proposition does not hold in general--not even when $\succ$ is an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E}$.

Other minor limitations, while evident, need not be discussed in detail. Clearly, the relation between normal refinements and their realizations in normal proof procedures needs to be explored in greater detail. The "structured" specification of search strategies outlined in §2.4.3 (Composition of enqueuing functions) is particularly tentative; however, a simple breadth-first search strategy would suffice for initial implementations of normal proof procedures.

The limitation to first-order syntax with typeless variables is a matter of initial convenience and simplicity of presentation rather than of basic barriers to extension; however, the basis for this assertion is only hinted at in §4.3.3.

5.3 Extensions and Applications

5.3.1 Preliminary Empirical Investigations

It is feasible and appropriate to begin an empirical investigation of normal proof procedures based on the performance evaluation framework of §B. Such an investigation will be based on (i)-(iii):

(i) A collection of benchmark application environments, each effectively represented by a finite (or at least decidable) axiom system and a finite weighted sample of refutation problems based on this axiom system.

(ii) An implemented refutation procedure with facilities for the specification of various normal refinements and (optionally) various search strategies.

(iii) Appropriate computational facilities and resources.

In order to facilitate comparison with previous and ongoing research, the sample problems should include problems previously attempted by experimental refutation procedures. Numerous problems in group theory and other equational theories have been previously investigated and would provide comparative information on the usefulness of ND($\mathcal{E}$,$\succ$). Problems in Integer Arithmetic have been undertaken by several proof procedures being used in program verification applications. Bledsoe et al have developed specialized proof procedures and applied them to problems in real analysis [7] and in general topology [8]. In view of my claim (§1.1.4) that many forms of structural heuristic knowledge can be formalized within normal refinements, it would also be useful to investigate application environments where a great deal of heuristic knowledge has been developed (e.g., sentence recognition in context-free languages), in order to see how much of this knowledge can be formalized without resorting to ad hoc methods.

The Interactive Theorem Prover of Luckham et al [1] provides an appropriate basis for initial implementations and investigations of normal proof procedures. This LISP-based system, based on pairwise resolution and paramodulation, includes numerous facilities for the specification of refinements by means of user-programmable pre-editing and post-editing strategies. Comparison with existing refinements

would be particularly simple, because many of these are built-in options of the Interactive Theorem Prover. This system is portable, and can be set up on most PDP-10 computer systems with sufficient storage capacity and a teletype or (preferably) graphic display terminal.

5.3.2 Investigation of Generalized $\mathcal{E}$-Resolution Refinements

This research has shown the need for an extension of $ND(\mathcal{E},\succ)$ havng a property called strong $\underline{\mathcal{E}}$-completeness (§3.3, Theorem 9). The extension eventually arrived at will probably realize an $\mathcal{E}$-resolution inference

$$\{B_i \vee q_i : i < n\} \vdash C \qquad (1)$$

by first finding a deduction $\underline{\mathcal{D}}(C')$ based on $\{B_i \vee q_i : i < n\} \cup \mathcal{E}$ in $ND(\mathcal{E},\succ)$. If $B_i = 0$ for each equation q_i, then $C' = C$. Otherwise, C' is converted to C by an "inversely" $\underline{\mathcal{E}}$-normal derivation based on variants of clauses in $\{B_i \vee q_i : i < n\}$. This derivation will replace only certain occurrences of terms in C' which are obtained from $(B_0 - C_0) \vee \cdots \vee (B_{n-1} - C_{n-1})$ by instantiation (where $C' = ((B_0 - C_0) \vee \cdots \vee (B_{n-1} - C_{n-1}))\sigma_{\underline{\mathcal{D}}(C')}$), and the ultimate (derived) replacement for each such term must be in $NF(\mathcal{E},\succ)$. The purpose of the derivation from C' to C is simply to "undo" any local identifications of terms in C' which may have been introduced by using equations q_i from non-unit clauses in the refutation of $\{C_i : i < n\} \cup \mathcal{E}$ from which $\underline{\mathcal{D}}(C')$ is obtained.

In the realm of non-unit (generalized) $\mathcal{E}$-resolution refinements, there are numerous unexplored alternatives to $HR(\mathcal{E},\succ,s)$. For instance, a linear (generalized) $\mathcal{E}$-resolution deduction is a (generalized) $\mathcal{E}$-resolution deduction $\underline{\mathcal{D}}$ wherein each non-initial

clause C has a near parent (one of its premises) and a set of far parents; each far parent (also a premise) is either an initial (input) clause or an ancestor of C in $\mathcal{D}$. The only class of linear generalized $\mathcal{E}$-resolution deductions heretofore investigated is essentially that in which the only inferences allowed are pairwise resolution and paramodulation. Perhaps the linear resolution experts will find it possible to relativize some of their refinements to generalized $\mathcal{E}$-resolution while retaining useful completeness and efficiency results such as those of Kowalski and Kuehner [43].

5.3.3 Formal Improvements

The definitional hierarchy developed in the course of this research could be usefully altered in several respects. Much conceptual clarity would be gained by formally requiring that no premise of a (generalized) $\mathcal{E}$-resolution inference be subsumed by a clause in $\mathcal{E}$. (Theorem A would then have to be restated: the negative clauses must be excluded from $\mathcal{E}$.)

The concept of a refutation procedure for Γ could be usefully generalized to incorporate the following paradigm:

(i) The procedure is activated with a set $\mathcal{C}$ of input clauses.

(ii) A Γ-refinement Δ to be used for the refutation of $\mathcal{C} \cup Ax(\Gamma)$ is selected by the procedure.

(iii) An enqueuing function (or search strategy) E is selected by the procedure for the search of Δ .

(iv) A complete deduction in $\Delta(= \Delta(\mathcal{C} \cup Ax(\Gamma))$ is generated as in Ref [E/Enq]($\mathcal{C}$).

Step (ii) might involve an analysis of $\mathcal{C} \cup Ax(\Gamma)$ by a procedure with special knowledge of extensions of $Ax(\Gamma)$ representing years of research by human specialists. The generalized concept differs from the present one in that Δ is selected after $\mathcal{C}$ (the particular problem) is known, and E is selected after the space to be searched (Δ) is known. The performance-evaluation measures for proof procedures developed in B remain applicable.

This strategy of transforming the search space (and search strategy) as a part of the problem-solving process has been anticipated and advocated by Simon [74] and Amarel [2].

5.3.4 Extensions to Type-Structured Logics

The following paragraphs outline a family of extensions of resolution-based calculi and proof procedures to the applied logical systems mentioned in §4.4. The basic motivation for these extensions is simplicity and naturalness in the formalization of various metalinguistic concepts such as application (of an operation or relation to an operand) and interpretation (of a constant or variable symbol).

Applicative systems. Conceptual and computational simplicity will be enhanced by the adaptation of a uniform representation for terms and atomic formulas based on the binary operation (__) known as application (§4.3.1). A vocabulary will consist of (__) plus a countably infinite set of atoms, which will be used as constants and variables denoting individuals, operations, and relations under various interpretations. Forms constitute the smallest class $\mathcal{F}_V$ of expressions which includes all atoms and all expressions (uv) where u and v are forms. The application of an "n-ary function" f to

arguments $t_1,\cdots,t_n$ may be represented either by the n-fold application $(ft_1\cdots t_n) =_{df} (\cdots(ft_1)\cdots t_n)$ or by an application $(f\ \langle t_1,\cdots,t_n\rangle)$ where $\langle t_1,\cdots,t_n\rangle$ is a defined form which represents the n-tuple $(t_1,\cdots,t_n)$. f represents an n-ary relation provided that $(ft_1\cdots t_n)$ (or $(f\ \langle t_1,\cdots t_n\rangle)$) denotes one of the designated truth values (0 for false, 1 for true) under the class of admissible interpretations.

Applicative logics will be formulated on the basis of four additional primitive concepts:

a) Equality, represented by an atomic constant $=$ and appropriate inference rules (essentially Rp) operating on Boolean forms (below):

$$[u=v] =_{df} ((=u)v)$$

b) Extensionality, represented by an atomic constant $\#$ (the discriminator or choice functor) and the extensionality axiom (a Boolean form):

$$(\#)\ [[x(\#xy) = y(\#xy)] = [x=y]]$$

Under each admissible interpretation wherein $u \neq v$, $(\#uv)$ is an object such that $(u(\#uv)) \neq (v(\#uv))$. Thus $[x(\#xy) = y(\#xy)]$ is true iff $[x=y]$ is true.

c) Functional abstraction, represented either by the constants I,K,S and corresponding axioms (§4.3.1), or by a primitive functional abstraction operation $(\lambda\_\_)$ in addition to $(\_\_)$, wherein (λxu) is a λ-form with bound variable x (an atom) and body u (a form or λ-form).

The former representation (wherein (λxu) is a defined form constructed from I,K,S and subterms of u) is algebraically cleaner and is useful in establishing formal results such as completeness and consistency. The latter (well known to be equivalent) is definitely superior for computational purposes. In either case, ((λxu)t) denotes u[t/x] provided that the type of t is a subtype of the type of x (see below).

d) Grammar, represented by a formal metalinguistic system which specifies

(i) A class of grammatical (admissible) interpretations of the primitive operations (__) and (optionally) (λ__) and atoms;

(ii) A class of grammatical forms, which have a denotation under each grammatical interpretation.

(iii) A class of Boolean forms, which denote a truth value (normally 0 or 1) under each grammatical interpretation. A calculus Γ for such a logic is over the class of Boolean forms--i.e., $Ax(\Gamma)$ is a class of Boolean forms, and if $\mathcal{B} \vdash_{\Gamma} C$ then $\mathcal{B} \cup \{C\}$ is a set of Boolean forms.

On the basis of the above, it is possible to formulate all of the usual logical concepts of first-order logic and of the theory of finite simple types [26,27,70]. The usual logical operators may be defined--e.g.,

$$\forall x\ B =_{df} [\lambda xB = \lambda xl]$$

or they may be introduced as additional primitives (depending upon whether the primary objective is conceptual simplicity or computational efficiency). In either case, a normal-form representation (clause form) can be defined for Boolean forms much as in first-order logic. The principal difference here is the necessity of deciding how to treat embedded Boolean forms such as (Cond[x=0] 1 [x · fact(x-1]) in the familiar Boolean form

$$[fact(x) = (Cond[x=0]\ 1\ [x \cdot fact(x-1)])]$$

(more commonly expressed as

$$[fact(x) = \underline{If}\ [x=0]\ \underline{then}\ 1\ \underline{else}\ x \cdot fact(x-1)]$$

Normal form logics will be based on the combinatory form of (c) (I,K, and S) and the normal form grammar, wherein the grammatical forms are essentially those wherein every subform has a strong normal form in the sense of combinatory logic [30]. Boolean forms in this logic consist essentially of those equations [u=v] wherein each subterm of u or v has a strong normal form. The "Russell paradox" form (cc) where c represents $\lambda x((xx)=0)$ is not Boolean in this grammar because it has no strong normal form. Thus, the fact that $(cc) =_{\mathcal{E}} [(cc) = 0]$ (where $\mathcal{E} = \{(I),(K),(S),(\#)\}$) does not render the logic inconsistent by identifying 0 and 1. (If we add [cc = 0] as an axiom then we derive [[cc = 0] = 0], [[0 = 0] = 0], and [1 = 0] --a contradiction. If we add [cc = 1] as an axiom then we derive [[cc = 0] = 1], [[1 = 0 = 1] --also a contradiction. These derivations will be admitted by any of the anticipated extensions of Cut, Rp , and $ND(\mathcal{E},>)$ to the present applicative framework.

The extension of $ND(\mathcal{E},\succ)$ for theorem-proving applications in normal form logics is particularly appealing. Even though the set of grammatical forms in this logic is only semi-decidable, the bottom-to-top $\underline{\underline{\mathcal{E}}}$-normal reductions performed by $ND(\mathcal{E},\succ)$ efficiently incorporate grammaticality proofs into proofs of theorems; every literal must be reduced to strong normal form before it can be used in Rp or Cut inferences involving other clauses.

Type systems. The grammaticality concepts in (d) can be formalized by the concept of a type system $\underline{\underline{\mathcal{E}}} = (\mathcal{E},\succ,V_C)$ satisfying (a)-(c):

(a) $\mathcal{E}$ is a decidable set of forms in $\mathcal{F}_V$.

(b) $\succeq$ is a monotone quasi-ordering for $\mathcal{F}_V$.

(c) V_C is a decidable set of constant atoms in V, and $\{=,\#,\text{Null},\text{Nil},\text{Bool},\text{Any}\} \subseteq V_C$.

(d) Each member of $\mathcal{E}$ contains at most one free variable (in $V - V_C$).

Additional constraints on $\underline{\underline{\mathcal{E}}}$ ensure that $\underline{\underline{\mathcal{E}}}$ defines a unique type structure $T(\underline{\underline{\mathcal{E}}}) = (T, \cap, \text{Null}, \text{Any}, \text{Nil}, \text{Bool}, \text{Co}, \tau)$ satisfying (e)-(h), wherein

$$(\alpha \subset \beta) =_{df} (\alpha \cap \beta = \alpha) \wedge (\alpha \cap \beta \neq \beta) ,$$

$$(\alpha \subseteq \beta) =_{df} (\alpha \subset \beta) \vee (\alpha = \beta) , \quad \text{and}$$

$$V_I =_{df} V - V_C .$$

(e) T is a decidable set of objects called types.

(f) $\cap$ is a computable binary operation on T such that $(T,\cap,\text{Null},\text{Any})$ is a dual semi-lattice with zero-element

Null, unit element Any, and an atom Nil, where $\text{Null} \subset \text{Nil} \subset \text{Bool}$.

(g) $\text{Co}: T \times T \to T$, the <u>codomain</u> function, is a computable operation on T such that

(i) $\text{Co}(\text{Null},\alpha) = \text{Co}(\alpha,\text{Null}) = \text{Null}$;

(ii) If $\text{Null} \subset \alpha' \leq \alpha$ and $\text{Null} \subset \beta' \leq \beta$, then $\text{Null} \subset \text{Co}(\alpha',\beta') \leq \text{Co}(\alpha,\beta)$; and

(iii) For each type α in T, $\text{Co}(\alpha,\text{Any}) = \text{Bool}$.

(h) $\tau: \mathscr{F}_V \to T$ is a computable function such that

(i) $\tau(u) \neq \text{Null}$ and $\tau(u,v) \leq \text{Co}(\tau(u),\tau(v))$ $(u,v \in \mathscr{F}_V)$;

(ii) $\tau(u) = \text{Bool}$, for each form u in $\mathcal{E}$;

(iii) $\mathcal{E}$ contains $(\tau(v)v)$ for each atom v in V;

(iv) if $\mathcal{E}$ contains u where a variable x occurs free in u, then $u = (\tau(x)x)$;

(v) if $\tau(t) \leq \tau(x)$ then $\tau((\lambda xu)t) = \tau(u[t/x])$; and

(vi) if $\tau(t) \cap \tau(x) = \text{Null}$ then $\tau((\lambda xu)t) = \text{Nil}$.

<u>Notation</u>. Given a homomorphism ψ on $\underline{\underline{\mathscr{F}}}_V = (\mathscr{F}_V,(\_\_))$, let $u_\psi =_{df} \psi(u)$ $(u \in \mathscr{F}_V)$.

Let $\underline{\underline{A}}$ be a mapping on $V_C \cup \{(\_\_),(\lambda\_\_)\}$ such that $\text{Any}_{\underline{\underline{A}}}$ is a set and $(\_\_)_{\underline{\underline{A}}}$ is a binary operation on $\text{Any}_{\underline{\underline{A}}}$. Let $V[\underline{\underline{A}}] = V \cup \text{Any}_{\underline{\underline{A}}}$. Extend $\underline{\underline{A}}$ to constant forms (i.e., forms containing no free variables) in $\mathscr{F}_{V[\underline{\underline{A}}]}$ as follows:

(i) $a_{\underline{\underline{A}}} = a$ $(a \in \text{Any}_{\underline{\underline{A}}})$;

(ii) $(uv)_{\underline{\underline{A}}} = (u_{\underline{\underline{A}}} v_{\underline{\underline{A}}})_{\underline{\underline{A}}}$; and

$$\text{(iii)}\quad ((\lambda xu)_{\underline{A}}b)_{\underline{A}} = \begin{cases} u[b/x]\underline{A}, & \text{if } (\tau(x)_{\underline{A}}b)_{\underline{A}} = 1 \quad ; \\ 0, & \text{otherwise } (b \in \text{Any}_{\underline{A}}). \end{cases}$$

For each type α in T, $\alpha_{\underline{A}}$ will normally be identified with the set $\{a \in \text{Any}_{\underline{A}}: (\alpha_{\underline{A}}a)_{\underline{A}} = 1\}$.

An $\underline{\mathcal{E}}$-structure is a mapping $\underline{A}$ on $V_C \cup \{(\_\_),(\lambda\_\_)\}$ satisfying (a)-(f):

(a) $(\_\_)_{\underline{A}}$ is a binary operation on $\text{Any}_{\underline{A}}$ such that if $b \in \beta_{\underline{A}}$ and $c \in \gamma_{\underline{A}}$ then $(bc)_{\underline{A}} \in \text{Co}(\beta,\gamma)_{\underline{A}}$ $(b,c \in \text{Any}_{\underline{A}})$

(b) $[b=c]_{\underline{A}} = \begin{cases} 1, & \text{if } b = c \ ; \\ 0, & \text{otherwise } (b,c \in \text{Any}_{\underline{A}}); \end{cases}$

(c) $(b(\#bc))_{\underline{A}} = (c(\#bc))_{\underline{A}}$;

(d) $(\alpha \cap \beta)_{\underline{A}} \leq \alpha_{\underline{A}} \cap \beta_{\underline{A}}$ $(\alpha,\beta \in T)$;

(e) $\text{Null}_{\underline{A}} = 0$; and

(f) $u_{\underline{A}} = 1$, for each constant form (or axiom) in $\mathcal{E}$.

An assignment in $\underline{A}$ is a mapping $\theta: V_I \to V_I \cup \text{Any}_{\underline{A}}$ such that

(i) if $x\theta \in V_I$ then $x\theta = x$; and

(ii) if $x\theta \in \text{Any}_{\underline{A}}$ then $x\theta \in \tau(x)_{\underline{A}}$.

If $x_i\theta = a_i$ $(i=0,\ldots,n-1)$ and $x\theta = x (x \notin \{x_i: i<n\})$ then θ is denoted by $[a_i/x_i: i<n]$. If $x\theta \in \text{Any}_{\underline{A}}$ $(x \in V_I)$, then θ is a total assignment in $\underline{A}$.

An $\underline{\mathcal{E}}$-interpretation is a mapping $\phi = \underline{A}\cdot\theta$ where $\underline{A}$ is an $\underline{\mathcal{E}}$-structure and θ is a total interpretation in $\underline{A}$.

Remarks.

1. A consequence relation $\models_{\underline{\underline{\mathcal{E}}}}$ is defined in terms of the class of all $\underline{\underline{\mathcal{E}}}$-interpretations, much as in §1.2.7.

2. Given an $\underline{\underline{\mathcal{E}}}$-interpretation ϕ in $\underline{\underline{A}}$ and a Boolean form B , $(\forall x\ B)_\phi = 1$ iff $B[a/x]_\phi = 1$ for all a in $\tau(x)_{\underline{\underline{A}}}$. (Clearly this holds if $\forall x\ B =_{df} (\lambda x B = \lambda x 1)$.)

A unification algorithm mgu for finite sets of $\underline{\underline{\mathcal{E}}}$-grammatical forms is easily defined on the basis of a type structure $\underline{\underline{T}}_{\underline{\underline{\mathcal{E}}}}$ for $\underline{\underline{\mathcal{E}}}$: given a finite set $\{u_1,\cdots,u_n\}$ of $\underline{\underline{\mathcal{E}}}$-grammatical forms, $\text{mgu}\{u_1,\cdots,u_n\}$ is either nil or a substitution θ in $\Sigma_{\underline{\underline{\mathcal{E}}}}$ such that $u_1\theta = \cdots = u_n\theta$, an $\underline{\underline{\mathcal{E}}}$-grammatical form.

$$\Sigma_{\underline{\underline{\mathcal{E}}}} =_{df} \{\theta \in \Sigma_V:\ \tau(x\theta) \subseteq \tau(x)\ ,$$
$$=\theta\ =\ \underline{=}\ ,\ \text{and}$$
$$\#\theta = \#\}$$

Given variables x and y (variables of V being defined by $\underline{\underline{\mathcal{E}}}$), $\text{mgu}\{x,y\}$ is nil if $\tau(x) \cap \tau(y) = \text{Null}$; otherwise $\text{mgu}\{x,y\} = [z/x,z/y]$ where $\tau(z) = \tau(x) \cap \tau(y)$. If c is a constant atom and $\tau(c) \not\subseteq \tau(x)$ then $\text{mgu}\{c,x\} = \underline{\text{nil}}$; if $\tau(c) \subseteq \tau(x)$ then $\text{mgu}\{c,x\} = [c/x]$.

5.3.5 Type-Structured Procedural Logics

Current research on operating systems, robotic systems, and automatic programming is leading to the development of high-level programming languages with facilities for specifying and reasoning about concurrent and continuous processes [10,26]. It is becoming

increasingly necessary to develop logics and proof procedures which are well suited to the verification and synthesis of such processes. Procedural logic is an extension of first-order (or type-structured) predicate logic wherein forms may contain certain state variables and a structure may contain mappings, called states, from the state variables to their values in the structure. Modal quantifiers, expressing truth or falsity of Boolean forms under various states related to the current (or initial) state of the structure, may also be present. A procedural logic based on first-order logic is investigated by McCarthy and Hayes in [50].

For several programming languages currently under development [35,36,10] a rich concept of type structure such as that of §5.3.4 will clearly be a part of the language's procedural logic. The concept of a type-structured procedural logic outlined below is a syntactical and semantical extension of the type-structured logics described in §5.3.4 (as opposed to a mere axiomatization exercise within first-order or type-structured logic). The ultimate practicality of this approach depends upon the successful extension of normal refinements and proof procedures from first-order logics to the enriched logics.

A modal type system has the form $\underline{\underline{\mathcal{E}}} = (\mathcal{E},\succ,V_C,V_S)$ where

(i) $(\mathcal{E},\succ,V_C)$ is a type system (§5.3.4);

(ii) V_S, the set of state variables, is a finite subset of $V_I = V - V_C$;

(iii) T contains a type Σ, the type of states, such that $Co(\Sigma,\alpha) = \alpha(\alpha \in T)$, where $T(\underline{\underline{\mathcal{E}}}) = (T,\cap,Null,Any,Nil,Bool,$

Co,τ) as in §5.3.4; and

(iv) V_C contains a symbol σ_o, the initial or current state, such that $\tau(\sigma_o) = \Sigma$.

An $\underline{\underline{\mathcal{E}}}$-structure is a mapping $\underline{\underline{A}}$ on $V_C \cup \{(\_\_),(\lambda\_\_)\}$ such that

(i) $(\mathcal{E},\succ,V_C,V_S)$ is an $\underline{\underline{\mathcal{E}}}$-structure as in §5.3.4;

(ii) $\Sigma_{\underline{\underline{A}}}$ is a set of states $[a_i/x_i: i < n]$ in $\underline{\underline{A}}$, where $V_S = \{x_i: i < n\}$ and $(\tau(x_i)_{\underline{\underline{A}}}\ a_i)_{\underline{\underline{A}}} = 1$ $(i < n)$;

(iii) $(\sigma u)_{\underline{\underline{A}}} = [u\sigma]_{\underline{\underline{A}}}$ $(\sigma \in \Sigma_{\underline{\underline{A}}})$; and

(iv) $u_{\underline{\underline{A}}} = [u\sigma_{o_{\underline{\underline{A}}}}]_{\underline{\underline{A}}}$ $(u \in \mathcal{F}_{V[\underline{\underline{A}}]})$.

Observe that T may contain numerous state types $\alpha \subseteq \Sigma$ and that V_I may contain variables of type α. α may be formally represented as a defined type $\lambda\sigma(\sigma B)$ where B is a Boolean form containing state variables and $\tau(\sigma) = \Sigma$: $\lambda\sigma(\sigma B)_{\underline{\underline{A}}} = \{\sigma \in \Sigma_{\underline{\underline{A}}}: [B\sigma]_{\underline{\underline{A}}} = 1\}$.

Thus we may formalize the concept of an action as a form of type $[\alpha \to \beta]$ where $\alpha \subseteq \Sigma$, $\beta \subseteq \Sigma$, and $Co([\alpha \to \beta].\alpha) = \beta$ α is the precondition of the form, and β is the postcondition of the form. Actions in higher-level programming languages are expressed by assignment conditional actions, for-statements, while-statements, and blocks (grouped sequences of actions).

Mappings from states to other objects (value-blocks), mappings from objects to actions (procedures) and mappings from objects to value-blocks (operations) can be formalized similarly.

Synthesis and verification of these program objects will be facilitated by allowing a block π to contain, in addition to preconditions, other assertions such as the following "modal quantifier" assertions, which are relative to the computation sequence(s) defined by (π,A,σ_0):

Invariant(B): $(\sigma B)_{\underline{\underline{A}}} = 1$ for each state σ achievable from σ_0 in this block;

Achievable(B): $(\sigma B)_{\underline{\underline{A}}} = 1$ for some achievable state σ ; and

Inevitable(B): There exists no complete computation sequence from $(\pi,\underline{\underline{A}},\sigma_0)$ wherein B is false for each state.

The latter operator should be useful for reasoning about termination properties of program entities.

Refutation procedures for a type-structured procedural logic such as the above must include state-space generation and search procedures for evaluating modal formulas such as Invariant(B) or Achievable(B) , in addition to the usual facilities for reasoning within a state. For example, in order to prove Achievable(B) we would either construct an explicit derivation of a state σ from σ_0 by means of π (possibly a "nondeterministic" block with optional and concurrent actions) wherein $(\neg B)\sigma$ is inconsistent with other assertions which hold in σ . (σ may be a symbolic expression composed of actions applied to σ_0.)

5.3.6 Relevancy of Present Research .

The extension of normal refinements and proof procedures to the type-structured procedural logics outlined in §5.3.4-§5.3.5 is both complex and sketchy. Nevertheless, progress toward completing such an extension has been encouraging, and the advantages of extending the logic rather than axiomatizing the concepts of type and state in a standard first-order logic are becoming clearer.

The present investigation of normal refinements is relevant to the proposed extension in at least two respects. I pointed out in §4.3 that refinements of the form $ND(\mathcal{E},\succ)$ offer attractive alternatives to earlier (and I believe unfortunate) attempts to synthesize algebraic simplification and unification in theories containing many "reduction equations". Thus, much of the potential redundancy resulting from a "λ-calculus" formalism can be eliminated by extensions of refutation procedures based on $ND(\mathcal{E},\succ)$.

Moreover, the structure of normal refinements and proof procedures is well matched to the structure of axiom systems which arise from programs expressed in languages having built-in procedural logics. Consider, for example, block B_1 in the context

$$B_3: \left[\begin{array}{l} \mathcal{E}_3 \\ B_2: \left[\begin{array}{l} \mathcal{E}_2 \\ B_1: \left[\begin{array}{l} \mathcal{E}_1 \end{array} \right. \end{array} \right. \end{array} \right.$$

When B_1 is entered, the assertions of $\mathcal{E}_1$ are evaluated in the current state and appended to assertions obtained similarly from $\mathcal{E}_3$ and

$\mathcal{E}_2$. It seems natural to require that B_1 specify, either explicitly or by default, both a refinement Δ_1 and a search strategy E_1 to use when non-unit clauses of $\mathcal{E}_1$ are involved. Thus, whenever the refutation procedure is invoked by the current processor of B_3 (whether for verification, compilation, or decision-making during execution), a normal refinement $(\Delta_3 \cdot \Delta_2 \cdot \Delta_1 \cdot \Delta_0)^{\Delta_\mu}$ with a well-matched search strategy based on (E_3, E_2, E_1) is automatically available.

In conclusion, I see the structured design method developed in this report as a basic contribution to the development of powerful deductive problem-solving systems based on type-structured procedural logics for arbitrary axiomatizable problem domains. The present results, while only a small fragment of the extensions and applications outlined in §5.3, shed sufficient light on the remainder to support its feasibility and worthiness of further investigation.

APPENDICES

A. PROOFS OF PRINCIPAL LEMMAS

This appendix contains the proofs of Lemmas 1-12 as stated and used in §3. Definitions and results from §3 are mentioned without duplication below.

A.1 Proofs for $\mathcal{E}$-resolution Completeness Lemmas

Proof of Lemma 1 (by induction on $n = \kappa(\mathcal{C})$). Suppose without loss of generality that every proper subset of $\mathcal{C}$ is $\mathcal{E}$-satisfiable (whence $\mathcal{C}$ is finite by the Relative Compactness Theorem (§3.0)).

Suppose $n = 0$. Then $\mathcal{C} \cup \mathcal{E} \cup [x=x] \geq \{p,\tilde{q}\}$ where $r_s(p)$ is positive, $r_s(\tilde{q})$ is negative, and $p\theta =_{\mathcal{A}} q\theta$ where $\mathcal{A} = (\mathcal{C} \cup \mathcal{E} \cup [x=x]) \cap \mathcal{A}_V$. If p is an equation then assume without loss of generality that $p = [x=x]$. It follows that $(\mathcal{A} \cup \{p,\tilde{q}\} \vdash 0)$ is the unique inference in a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C}$.

Suppose $n > 0$ and Lemma 1 holds for all $\mathcal{C}$ such that $\kappa(\mathcal{C}) < n$. Suppose without loss of generality that each proper subset of $\mathcal{C}$ is $\mathcal{E}$-satisfiable.

Case 1: Each non-unit clause of $\mathcal{C}$ is r_s-negative. Then $\mathcal{C}$ contains a unique non-unit clause $\tilde{q}_1 \vee \cdots \vee \tilde{q}_n$. Let $\mathcal{C}' = \mathcal{C} - \{\tilde{q}_1 \vee \cdots \vee \tilde{q}_n\}$, and let $\mathcal{C}_i = \mathcal{C}' \cup \{\tilde{q}_i\}$. Then $\mathcal{C}_i$ is $\mathcal{E}$-contradictory and $\kappa(\mathcal{C}_i) = 0$. It follows by induction that there exists a hyper-$\underline{\mathcal{E}}$-resolution refutation $\underline{\mathcal{D}}_i(0)$ of $\mathcal{C}_i$ consisting of a single hyper-$\underline{\mathcal{E}}$-resolution inference $\mathcal{B}_i \cup \{\tilde{q}_i\} \vdash 0$. Therefore $\mathcal{C}$ is refuted by the single hyper-$\underline{\mathcal{E}}$-resolution inference

$$\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n \cup \{\tilde{q}_1 \vee \cdots \vee \tilde{q}_n\} \vdash 0.$$

Case 2: ¬(Case 1). Let $\underline{\mathcal{E}} = (\mathcal{E}, \succ, s)$ where $\succ$ is the ordering of $\mathcal{A}_V$ wherein equations precede other atoms. Let

$$\mathcal{A} = \{q: \mathcal{C} \text{ contains a non-unit clause } A \vee q \text{ where } q \text{ is } r_s\text{-positive}\} .$$

Let p be a literal of $\mathcal{A}$ such that $r_s(q) \geq r_s(p)$ $(q \in \mathcal{A})$, and let $A \vee p$ be a non-unit clause of $\mathcal{C}$ where $A \cap \{p\} = 0$. Let $\mathcal{C}' = \mathcal{C} - \{A \vee p\}$, $\mathcal{C}_A = \mathcal{C}' \cup \{A\}$, and $\mathcal{C}_p = \mathcal{C}' \cup \{p\}$. It follows by induction that $\mathcal{C}_A$ has a hyper-$\underline{\mathcal{E}}$-resolution refutation $\underline{\mathcal{D}}_A$ and $\mathcal{C}_p$ has a hyper-$\underline{\mathcal{E}}$-resolution refutation $\underline{\mathcal{D}}_p$. Let $\underline{\mathcal{D}}_{A \vee p}$ be the result of embedding $\underline{\mathcal{D}}_A$ in $\mathcal{C} \cup \mathcal{E}$. $\underline{\mathcal{D}}_{A \vee p}$ is a hyper-$\underline{\mathcal{E}}$-resolution refutation because p is r_s-positive and $r_s(p)$ is minimal (hence, "last to be selected") in $\{r_s(q): q \in \mathcal{A}\}$ with respect to $\succ$. Suppose $\underline{\mathcal{D}}_{A \vee p}$ is not a refutation. Then $\underline{\mathcal{D}}_{A \vee p} = \underline{\mathcal{D}}_{A \vee p}(p)$. Prefixing $\underline{\mathcal{D}}_{A \vee p}$ to $\underline{\mathcal{D}}_p$, we obtain a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C}$. ∎

Proof of Lemma 2 (by induction on $n = \kappa(\mathcal{C})$).

Suppose without loss of generality that every proper subset of $\mathcal{C}$ is $\mathcal{E}$-satisfiable (whence $\mathcal{C}$ is finite).

Suppose $n = 0$. It follows by Lemma 1 (Corollary) that $\mathcal{C}$ contains at most one r_s-negative literal, and if each clause (literal) of $\mathcal{C}$ is r_s-positive then $\mathcal{E}^+$ contains an r_s-negative clause $(\tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1})$ such that $\mathcal{C} \models_{\mathcal{E}} q_j$ $(j=0,\cdots,n-1)$.

Case 1: $\mathcal{C}$ contains $\tilde{q}$, an r_s-negative literal. Then $(\mathcal{C} \vdash 0)$ is the unique hyper-$\underline{\mathcal{E}}$-resolution in a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C}$, by minimality of $\mathcal{C}$.

Case 2: ¬(Case 1). Then $\mathcal{E}$ contains an r_s-negative clause B where $B\theta = \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}$ and $\mathcal{C} \models_{\mathcal{E}} \tilde{q}_i$ $(i=0,\ldots,n-1)$. Therefore $(\mathcal{C} \cup \{B\} \vdash 0)$ is the unique inference in a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C}$.

Suppose that $n > 0$ and Lemma 2 holds for all $\mathcal{C}$ such that $\kappa(\mathcal{C}) < n$.

Case 1: Each non-unit clause of $\mathcal{C}$ is r_s-negative. Then $\mathcal{C}$ contains a non-unit r_s-negative clause $q_0 \vee \cdots \vee q_{n-1}$. The conclusion (of Lemma 2) follows by induction and the proof (Case 1) of Lemma 1.

Case 2: ¬(Case 1). Define $\mathcal{A}$, p, A∨p, $\mathcal{C}_A$, $\mathcal{C}_p$, $\underline{\mathcal{D}}_A$, $\underline{\mathcal{D}}_p$, $\underline{\mathcal{D}}$ from $\mathcal{C}$, r_s, $\succ$ as in the proof (Case 2) of Lemma 1. That $\underline{\mathcal{D}}$ is a hyper-$\underline{\mathcal{E}}$-resolution refutation of $\mathcal{C}$ follows by the same argument.∎

Proof of Lemma 3. That (4) is a hyper-$\underline{\mathcal{E}}$-resolution inference follows by (i) and substitutivity of $\succ$. (If p_i' is $\succ$-maximal in $B_i' \vee p_i'$ then p_i must be $\succ$-maximal in $B_i \vee p_i$.) That $C[\eta \cdot \theta'/\theta] \leq C'$ follows by (ii).∎

Proof of Lemma 4. It suffices to verify (using (i)-(vii)) that $(\mathcal{B}_{\mu_k} \vdash C_k \vee \tilde{q}_0^k \vee \cdots \vee \tilde{q}_{\nu_k}^k)$ is an $\underline{\mathcal{E}}$-resolution inference. This is straightforward. That (5) is general follows by the definition of $\mathcal{B} = \mathcal{B}_0 \cup \ldots \cup \mathcal{B}_{n-1} \cup \{B_n \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1}\}$.∎

A.2 Proofs for Resolution Micro-Refinement Lemmas

Proof of Lemma 5. It is easily verified (by §1.2.8) that $s =_{\mathcal{E}} t$ iff there exists an $\mathcal{E}^*$-derivation from s to t. Given that u and v are constant terms, we can instantiate each free variable in an $\mathcal{E}^*$-derivation from u to v by a constant atom e in V_F^o, obtaining an $\mathcal{E}^+$-derivation from u to v. ∎

Proof of Lemma 6. Notice that the set $(\hat{\mathcal{E}}^+)$, as opposed to $(\hat{\mathcal{E}})^+$, is referred to in Lemma 6. For each pair $[s=t_1],[s=t_2]$ of distinct equations in $\mathcal{E}^+$, at most one may appear in $\mathcal{E}^+$, because either $t_1 \succ t_2$ or $t_2 \succ t_1$. In view of the definition of $\Longrightarrow_{\hat{\mathcal{E}}^+}$, it is clear that there exists at most one $v \in NF(\mathcal{E},\succ)$ such that $u \Longrightarrow_{\hat{\mathcal{E}}^+} v$. However, there exists at least one by monotonicity of $\succ$ and the descending chain condition satisfied by $\succ$. ∎

Proof of Lemma 7. Let $\delta(u,v)$ be the minimal length of an $\mathcal{E}_\beta$-derivation sequence from u to v. If $\delta(u,v) = 0$ then $\hat{u} = \hat{v}$ because $u = v$.

Suppose $\delta(u,v) = n > 0$ and Lemma 7 holds for all u,v such that $\delta(u,v) < n$. Then $u = u'[s]$ where $\mathcal{E}_\beta$ contains $[s=t]$ or $[t=s]$ such that $\delta(u'[t],v) = n-1$. It follows by the induction hypothesis that $\widehat{u'[t]} = \hat{v}$. Let $\mathcal{D}'(P\hat{v})$ be a realization for $(u'[t]) \Longrightarrow^*_{\hat{\mathcal{E}}^+} \hat{v})$.

Obtain $\underline{\underline{\mathcal{D}}}(P\hat{v})$ realizing $(u \overset{*}{\Longrightarrow}_{\underline{\underline{\hat{\mathcal{E}}^+}}} \hat{v})$ from $\underline{\underline{\mathcal{D}}}'(P\hat{v})$ as follows. Replace [t] by [s] in Pu'[t] and its descendants in $\underline{\underline{\mathcal{D}}}'(P\hat{v})$, until all occurrences preceding [s] in one of these descendants have been reduced to $\underline{\underline{\mathcal{E}}}$-normal form. At this point we use the fact that $\hat{s} = \hat{t}$, reducing [s] to $[\hat{t}]$. The containing literal $Pu''[\hat{t}]$ is in $\underline{\underline{\mathcal{D}}}'(P\hat{v})$. $\underline{\underline{\mathcal{D}}}(P\hat{v})$ is obtained by suffixing the $\underline{\underline{\mathcal{E}}}$-normal reduction from $Pu''[\hat{t}]$ to $P\hat{v}$. ∎

Proof of Lemma 8. It suffices to prove that $\hat{u\theta} = \hat{v\theta}$ for each equation [u=v] in $\mathcal{E}$ such that $[u=v] \notin S[s=t]$ ($[s=t] \in \mathcal{E} - [u-v]^{\sim}$), $u\theta \succ v\theta$, and $[u\theta = v\theta] \in \mathcal{E}^+$.

Let $u\theta = s_\beta$, and suppose for the induction hypothesis that we have proven $\hat{u\theta} = \hat{v\theta}$ for all u,v,θ such that $[u-v] \notin S[s=t]$ ($[s=t] \in \mathcal{E} - [u-v]^{\sim}$), $u\theta \succ v\theta$, and $[u\theta = v\theta] \in \mathcal{E}_\beta$. It follows that $\hat{s} = \hat{t}$ for each equation $[s=t] \in \mathcal{E}_\beta$.

Case 1: Every proper subterm of $u\theta$ is in $NF(\mathcal{E}^+,\succ)$, and $v\theta$ is the first member of $\{v': [u\theta = v'] \in \mathcal{E}^+\}$. Then $u\theta \Longrightarrow_{\underline{\underline{\hat{\mathcal{E}}^+}}} v\theta \overset{*}{\Longrightarrow}_{\underline{\underline{\hat{\mathcal{E}}^+}}} v\theta$, whence $u\theta = v\theta$ by Lemma 6.

Case 2: Every proper subterm $u\theta$ is in $NF(\mathcal{E}^+,\succ)$ and $s\theta' = u\theta \succ v\theta \succ t\theta'$ where $\mathcal{E}^{\sim}$ contains [s=t], sharing no variables with [u=v], such that $[s=t] \notin S[s'=t']$ ($[s'=t'] \in \mathcal{E} - [s=t]^{\sim}$), either $x\theta = x$ or $x\theta' = x$ $(x \in V_I)$, and $t\theta'$ is the first member of $\{v': [u\theta = v'] \in \mathcal{E}^+\}$.

Suppose $u \in V_I$. It follows by normality of $\underline{\underline{\mathcal{E}}}$ that $\mathcal{E}$ contains [x=y] where $x,y \in V_I$, whence $\mathcal{E}^+$ includes

$\{u\theta = e],[v\theta = e]\theta\}$ (e being the first constant term). Consequently, $\widehat{u\theta} = \widehat{v\theta}$.

Suppose $u \notin V_I$, and let $\eta = mgu\{u,s\}$. Then η divides $\theta\cdot\theta'$, whence each proper subterm of $u\eta$ or of $s\eta$ is in $NF(\mathcal{E}^*,\succ)$ (because $u\theta\cdot\theta' = u\theta = s\theta\cdot\theta'$). It follows by the definition of closed ((ii) in §3.2.2) and minimality of $t\theta'$ that $t\eta \in NF(\mathcal{E},\succ)$. It follows by (iii) in §3.2.2 that

$$[t\eta = v\eta] \underset{\underline{\underline{\mathcal{E}}}}{\overset{*}{\rightarrow}} [u'' = v''] \qquad (1)$$

where $(\mathcal{E} \cup [x\text{-}x])^*$ contains $[u'' = v'']$ or $[v'' = u'']$.

Let $\underline{\underline{\mathcal{D}}}'$ be a general realization for (1), and let $\sigma = [\theta\cdot\theta'/\eta]$. Evidently there exists a simplest substitution σ' such that σ divides σ' and, for each inference

$$\{[s_i=t_i],q_i[r_i]\} \vdash q_i[t_i]\theta_i$$

in $\underline{\underline{\mathcal{D}}}'$, $r_i\sigma = r_i\sigma' = s_i\sigma' \succ t_i\sigma'$. Therefore $[s_i=t_i]\sigma' \in \mathcal{E}_\beta$ because $u\theta \succ t\eta\sigma \succ s_i\sigma' \succeq t_i\sigma'$. Moreover, either $u''\sigma' = v''\sigma'$ or $\mathcal{E}_\beta$ contains one of $[u''=v'']\sigma'$, $[v''=u'']\sigma'$, because $u\theta \succ t\theta' \succeq u''$ and $u\theta \succ v\theta \succeq v''\sigma'$. Thus, $\underline{\underline{\mathcal{D}}}'\sigma'$ shows that $t\theta' =_{\mathcal{E}_\beta} v\theta$.

It follows by Lemma 7 that $\hat{t}\theta' = \hat{v}\theta$. Let $\underline{\underline{\mathcal{D}}}''$ be a realization for $((Pt\theta') \underset{\underline{\underline{\mathcal{E}}}^+}{\overset{*}{\Rightarrow}} (Pv\theta))$, and let $\underline{\underline{\mathcal{D}}}$ be the result of prefixing

$$\{[s\theta'=t\theta'],(Pu\theta)\} \vdash (Pt\theta')$$

to $\underline{\underline{\mathcal{D}}}''$. Then $\underline{\underline{\mathcal{D}}}$ realizes $(u\theta \underset{\underline{\underline{\mathcal{E}}}^+}{\overset{\frown}{\Rightarrow}} v\theta)$, whence $\hat{u}\theta = \hat{v}\theta$.

Case 3: u contains x where $x\theta \notin NF(\mathcal{E}^*,\succ)$, and every proper subterm of $u[\hat{x\theta}/x: x\theta \neq x]$ is in $NF(\mathcal{E}^*,\succ)$.

Let $\tau = [\hat{x\theta}/x: x\theta \neq x]$. It follows that $u\theta \succ u\tau$, $v\theta \succeq v\tau$, and $\mathcal{E}_\beta$ contains either $[u=v]\tau$ or $[v=u]\tau$, whence $\hat{u\tau} = \hat{v\tau}$ by the induction hypothesis. Thus $\hat{u\theta} = \hat{v\theta}$ because $\hat{u\theta} = \hat{u\tau}$ and $\hat{v\tau} = \hat{v\theta}$ (by induction hypothesis).

Case 4: $\theta \in \Sigma_{\underline{\mathcal{E}}}$, and some proper subterm of $u\theta$ is not in $NF(\mathcal{E}^*,\succ)$. It follows that $u = u'[r]$ where $r \notin V_I$ and $t'\theta \in NF(\mathcal{E},\succ)$ for every term t' which occurs in $u'[r]$ before the position of $[r]$. Moreover, $\mathcal{E}^{\cup}$ contains $[s=t]$ sharing no variables with $['[r] = v]$ such that $r\theta = s\theta' \succ t\theta'$ and $t\theta'$ is first in $\{[s\theta' = t']: [s\theta' = t'] \in \mathcal{E}^*\}$, for some θ' such that $x\theta = x$ or $x\theta' = x$ $(x \in V_I)$. Let $\eta = mgu\{r,s\}$. It follows by (iii) in §3.2.2 that

$$[u'[t]\eta = v\eta] \rightarrow^*_{\underline{\mathcal{E}}} [u''=v''] \qquad (2)$$

where $(\mathcal{E} \cup [x=x])^*$ contains $[u''=v'']$ or $[v''=u'']$.

Let $\underline{\mathcal{D}}'$ be a general realization for (2), and define σ,σ' from θ,θ',η, and $\underline{\mathcal{D}}'$ as in Case 2.

It follows by essentially the same argument as in Case 2 that $u'[t]\eta\sigma =_{\mathcal{E}_\beta} v\theta$, whence $\widehat{u'[t]\eta\sigma} = \hat{v\theta}$ by Lemma 7. Let $\underline{\mathcal{D}}''(P\hat{v\theta})$ be a realization for $(u'[t]\eta\sigma \Rightarrow^*_{\underline{\hat{\mathcal{E}}}^+} \hat{v\theta})$. By prefixing the inference

$$\{[s=t]\theta', Pu'[r]\theta\} \vdash Pu'[t]\eta\sigma ,$$

we obtain a realization for $u\theta \Rightarrow^*_{\underline{\hat{\mathcal{E}}}^+} \hat{v\theta}$, whence $\hat{u\theta} = \hat{v\theta}$.∎

Proof of Lemma 9. If $u\theta = \overline{u\theta}$ then the trivial deduction $(\{Pu\},0)$ satisfies the conclusion. Suppose for the induction hypothesis that $u\theta \succ \overline{u\theta}$ and the lemma holds (with u' for u, θ' for θ for all u',θ' such that $u'\theta' \in [u\theta]_{\mathcal{E}}$ and $u\theta \succ u'\theta'$.

Case 1: $u \in NF(\mathcal{E},\succ)$. Let $\mathcal{D}'$ be the realization for $(Pu\theta \Rightarrow^{*}_{\hat{\mathcal{E}}+} P\overline{u\theta})$ guaranteed by Theorem 5. Let r' be the term in $u\theta$ replaced by the first inference in $\mathcal{D}'$. r' does not occur in $x\theta$ for any x in V_I, because $r' \notin NF(\mathcal{E}^{+},\succ)$ and $x\theta \in NF(\mathcal{E}^{+},\succ)$. Therefore the first inference of $\mathcal{D}'$ has the form

$$(\{[s=t]_\tau, Pu'[r]\theta\} \vdash Pu'[t_\tau]\theta) \tag{3}$$

where

(i) $[s=t] \in \mathcal{E}^{\sim}$ and $[s=t]$ shares no variables with u;

(ii) $\tau = [x\tau/x: x \text{ occurs in } [s=t]] \in \Sigma_{\mathcal{E}}$;

(iii) $r' = r\theta = s\tau \succ t\tau$; and

(iv) $u = u'[r]$ where $r \notin V_I$.

Observe that $\tau\cdot\theta = \theta\cdot\tau$ because of (i) and (ii).

The above conditions define an $Rp_{\mathcal{E}}$-inference

$$(\{[s=t], Pu'[r]\} \vdash Pu'[t]\eta) \tag{4}$$

where η divides $\theta\cdot\tau$, $[\theta\cdot\tau/\eta] \in \Sigma_{\mathcal{E}}$, and

$$\begin{aligned} u'[r]\theta &\succ u'[t_\tau]\theta \\ &= u'[t]\theta\cdot\tau \\ &= (u'[t]\eta)\,[\theta\cdot\tau/\eta]. \end{aligned}$$

It follows by the induction hypothesis that there exists an $\mathcal{E}$-normal

derivation $\underline{\mathcal{D}}_1$ from $Pu'[t]\eta$ to Pv where Pv subsumes $\overline{Pu\theta}$, and

$$[x\sigma_{\underline{\mathcal{D}}_1}/x\text{: } x \text{ occurs in } u'[t]\eta] \text{ divides } [\theta\cdot\tau/\eta]. \tag{5}$$

Obtain $\underline{\mathcal{D}}$ from $\underline{\mathcal{D}}_1$ by prefixing (4). $\underline{\mathcal{D}}$ is an $\underline{\mathcal{E}}$-normal derivation from $\mathcal{E} \cup \{Pu\}$ to Pv, and $\sigma_{\underline{\mathcal{D}}} = \eta\cdot\sigma_{\underline{\mathcal{D}}_1}$.

Case 1 is completed by showing that $[x\sigma_{\underline{\mathcal{D}}}/x\text{: } x \text{ occurs in } u]$ divides θ. It follows from (5) that

$$\eta\cdot[x\sigma_{\underline{\mathcal{D}}_1}/x\text{: } x \text{ occurs in } u'[t]\eta] \text{ divides } \theta\cdot\tau \tag{6}$$

(In general, if σ divides τ then $\eta\sigma$ divides $\eta\tau$ $(\eta \in \Sigma_V)$.) Now

$$\begin{aligned}
&\eta \cdot [x\sigma_{\underline{\mathcal{D}}_1}/x\text{: } x \text{ occurs in } u'[t]\eta] \\
&\quad = [x\sigma_{\underline{\mathcal{D}}}/x\text{: } x\eta \neq x \text{ or } x \text{ occurs in } u'[t]\eta] \\
&\quad = [x\sigma_{\underline{\mathcal{D}}}/x\text{: } x \text{ occurs in } [s=t] \text{ or in } u] .
\end{aligned}$$

Thus,

$$[x\sigma_{\underline{\mathcal{D}}}/x\text{: } x \text{ occurs in } [s=t] \text{ or in } u] \text{ divides } \theta\cdot\tau \tag{7}$$

Since $[s=t]$ shares no variables with $u'[r]$ and τ acts only on variables of $[s=t]$, it follows from (7) that

$$[x\sigma_{\underline{\mathcal{D}}}/x\text{: } x \text{ occurs in } u] \text{ divides } \theta .$$

Case 2: $u \notin NF(\mathcal{E},\succ)$. Let $\underline{\mathcal{D}}_0$ be a realization for $(Pu) \Rightarrow_{\underline{\mathcal{E}}} (Pu_1)$ where $u_1 \in NF(\mathcal{E},\succ)$. It follows that $u \succ u_1$, whence $u\theta \succ u_1\theta$. It follows by the induction hypothesis that there exists an $\underline{\mathcal{E}}$-normal derivation $\underline{\mathcal{D}}_1$ from Pu_1 to Pv where Pv subsumes $\overline{Pu\theta}$

and $[x\sigma_{\mathscr{D}_1}/x\text{: } x \text{ occurs in } u_1]$ divides θ. Let $\mathscr{D}$ be the result of prefixing $\mathscr{D}_0$ to $\mathscr{D}_1$, so that $\sigma_{\mathscr{D}} = \sigma_{\mathscr{D}_0} \cdot \sigma_{\mathscr{D}_1}$. $\mathscr{D}$ is clearly an $\mathscr{E}$-normal derivation. Observe that

(i) $x\sigma_{\mathscr{D}_0} = x(x \text{ in } u)$;

(ii) $\{x\text{: } x \text{ occurs in } u\} \supseteq \{x\text{: } x \text{ occurs in } u_1\}$; and

(iii) $x\sigma_{\mathscr{D}_1} = x\ (x \text{ in } u \text{ but } x \text{ not in } u_1)$;

whence

$$[x\sigma_{\mathscr{D}}/x\text{: } x \text{ occurs in } u = [x\sigma_{\mathscr{D}_1}/x\text{: } x \text{ occurs in } u_1] \qquad (8)$$

Thus, $[x\sigma_{\mathscr{D}}/x\text{: } x \text{ occurs in } u]$ divides θ by the induction hypothesis.∎

A.3 Proofs for Normal Composition Lemmas

Proof of Lemma 10. Let $\mathscr{E} = (\mathcal{E}, \succ, s)$, and consider an $\mathscr{E}$-resolution inference

$$(\{B_i' \vee q_i'\text{: } i<n\} \vdash (B_0'\theta' - q_0'\theta') \vee \cdots \vee (B_{n-1}'\theta' - q_{n-1}'\theta'))$$

It suffices to observe that if $(B_i \vee q_i)\eta \subseteq B_i' \vee q_i'$, $q_i\eta = q_i'$, $B_i \vee q_i$ does not subsume $(B_0'\theta' - q_0'\theta') \vee \cdots \vee (B_{n-1}'\theta' - q_{n-1}'\theta')$ $(i=0,\cdots,n-1)$, and θ is a divisor of $\eta\cdot\theta'$ such that

(i) $\{q_i\theta\text{: } i<n\}$ is an $\mathcal{E}$-contradiction; and

(ii) if $q\eta\theta' = q_i'\theta'$ then $q\theta = q_i\theta$ $(q \in B_i)$ $(i=0,\cdots,n-1)$; then

$$(\{B_i \vee q_i\text{: } i<n\} \vdash (B_0\theta - q_0\theta) \vee \cdots \vee (B_{n-1}\theta - q_{n-1}\theta))$$

is an $\underline{\underline{\mathcal{E}}}'$-resolution inference whose conclusion subsumes $(B_0'\theta' - q_0'\theta') \vee \cdots \vee (B_{n-1}'\theta' - q_{n-1}'\theta')$, where $\underline{\underline{\mathcal{E}}}' = (\mathcal{E},\succ,s')$, $r_{s'} = r_s$, and $s'(B_{n-1} \vee q_{n-1}) = q_{n-1}$ (where $s(B_{n-1}' \vee q_{n-1}') = q_{n-1}'$). This follows by substitutivity of $\succ$ and definition of $\overline{HR}(\mathcal{E},\succ,s)$.∎

Proof of Lemma 11. It suffices to consider a general deduction in $\overline{HR}(\mathcal{E},\succ,s)$ of the form

$$
\begin{array}{ll}
\mathcal{B}_0 & C_0 \vee \tilde{q}_0 \vee \cdots \vee \tilde{q}_{n-1} \\
 & \quad | \\
 & C_1 \vee \tilde{q}_1\sigma_1 \vee \cdots \vee \tilde{q}_{n-1}\sigma_1 \\
 & \quad \vdots \\
\mathcal{B}_{n-1} & C_{n-1} \vee \tilde{q}_{n-1}\sigma_{n-2} \\
 & \quad | \\
 & C_n
\end{array}
\qquad (9)
$$

where

(i) $(\mathcal{B}_k \cup \{C_k \vee \tilde{q}_k\sigma_k \vee \cdots \vee \tilde{q}_{n-1}\sigma_k\} \vdash C_{k+1} \vee \tilde{q}_{k+1}\sigma_{k+1} \vee \cdots \vee \tilde{q}_{n-1}\sigma_{k+1})$ is a general $(\mathcal{E},\succ,s_k)$-resolution inference with induced substitution θ_k, where $r_{s_k} = r_s$ $(k=0,\cdots,n-1)$;

(ii) $\sigma_0 = \varepsilon$ and $\sigma_{k+1} = \sigma_k \cdot \theta_k$ $(k=0,\cdots,n-1)$; and

(iii) C_n is r_s-positive or empty.

(Each r_s-positive or empty clause of $\underline{\underline{\mathcal{D}}}'$ is either in Base($\underline{\underline{\mathcal{D}}}'$) or the conclusion of some such deduction.) Using essentially the same argument as for Lemma 4, we can construct a "permutation" of (9), having the same conclusion, which is in $HR(\mathcal{E},\succ,s)$.∎

Proof of Lemma 12. Observe that $x\theta_k \geq x\theta_{k+1}$ $(k \in N)$. Therefore $x\theta_{n_x+k} = x\theta_{n_x}$ $(k \in N)$ for some $n_x > 0$, because otherwise $>$ has an infinite descending chain. Let $n = \max\{n_x: x\theta_o \neq x\}$. Then $x\theta_m = x\theta_n$ $(x \in V_I)$ for all $m \geq n$.∎

B. Utility-Measures for Refinements

> The key question in the development of automatic proof procedures is efficiency, but it is rather surprising how little serious discussion and analysis of it there has been . . .. In this field there has also been a great deal of discussion of the relative merits of 'complete' or 'incomplete' or 'heuristic' procedures, often without too clear notions of what these are and certainly seldom with any serious consideration of how their different characteristics affect their effectiveness in finding proofs.
>
> --- B. Meltzer[1]

Utility of a refinement is a function of expected (or "average") performance of a proof procedure which uses it (in a given application environment ($\mathcal{U}_{\mathcal{E}}$,μ) (§1.1.1). This appendix provides a basis for evaluating refinements to be used by the refutation procedure Ref (§1.3.6).

The behavior of Ref on $\mathcal{C}$ is essentially a function of three parameters:

(i) Δ , a Γ-refinement defining the search-space of Ref($\mathcal{C}$);

(ii) Enq, an enqueuing function representing a search strategy for generating new inferences on the basis of the current deduction; and

(iii) Subsume, a "deletion strategy" whereby certain theorems are deemed unnecessary for completion of the current deduction.

---

[1] Prolegomena to a Theory of Efficiency of Proof Procedures, in Artificial Intelligence and Heuristic Programming, ed. N. V. Findler and Bernard Meltzer (American Elsevier Publ. Co., Inc., New York, 1971).

Here as in §1.3.6, however, Subsume is assumed to be a fixed procedure such that Subsume(A,R) simply inserts A in R ; most subsumption- and tautology-deletion strategies can be incorporated into the Γ-refinements under investigation, in the sense that a subsumed or otherwise deletable clause is not used in subsequent inferences.

In addition to providing a formal basis for investigating relations between refinements and (expected) performance of refutation procedures which use them, §B.1 also provides a basis for investigating relations between search strategies and (expected) performance of refutation procedures which use them.

B.1 A Framework for Performance Evaluation

A performance-evaluation system for Ref has the following constituents:

(a) a class $\{\Delta_i : i \in N\}$ of Γ-refinements;

(b) a class $\{E_i : i \in N\}$ of fair enqueuing functions;

(c) an application environment $(\mathcal{U}_\mathcal{E}, \mu)$ where $\mathcal{E} = Ax(\Gamma)$; and

(d) a cost functional $K: N \times N \times \mathcal{U}_\mathcal{E} \to R_+^\infty$ (= extended positive reals) such that $K_j^i(\mathcal{C}) < \infty$ iff $\underline{\underline{\Pi}}_j^i(\mathcal{C})$ is finite, where $\underline{\underline{\Pi}}_j^i =_{df} \underline{\underline{Ref}}[\Delta_i/\Delta\ ,\ E_j/Enq]$.

Recall that μ is a probability measure on $\mathcal{U}_\mathcal{E}$ (§1.1.1), and that $\mu(\mathcal{U})$ is intended to be the expectation that Ref will be applied to some clause-set $\mathcal{C}$ in $\mathcal{U}$. $K_j^i(\mathcal{C})$ is intended to represent the computational effort (a function of time and storage space) required to compute $\underline{\underline{\Pi}}_j^i(\mathcal{C})$ using $\underline{\underline{\Pi}}_j^i$.

Expected values. A number of real-valued performance measures for $\underline{\underline{\Pi}}^i_j$ on $\mathcal{U}_\mathcal{E}$ are defined below. Given $f: \mathcal{U}_\mathcal{E} \to R$, the expected value of f on $\mathcal{U}_\mathcal{E}$ is given by $\int f\, d\mu$. In the event that μ is a "finite weighted sample" measure such that $\mu(\mathcal{U}) = \sum(\omega_i: \mathcal{C}_i \in \mathcal{U})$ where $(\sum_{i=1}^{n} \omega_i) = 1$, $\int f\, d\mu$ is just the weighted average value $(\sum_{i=1}^{n} f(\mathcal{C}_i)\cdot\omega_i)$ of f on the sample space $\{\mathcal{C}_1, \cdots, \mathcal{C}_n\}$.

Remark. Questions of when $\int f\, d\mu$ is defined are not discussed below, partly because $\int f\, d\mu$ is always defined when μ is a finite weighted sample measure. In general there may well be interesting performance measures f, such as cost, for which f is measurable and $\int f\, d\mu$ (expected cost) is ∞.

Cost of deductions. Define $\overline{K}$ on $\bigcup \{\Delta_i: i \in N\}$ by

$$\overline{K}(\underline{\underline{\mathcal{D}}}) =_{df} \min\{K^i_j(\mathrm{Base}(\underline{\underline{\mathcal{D}}})): \underline{\underline{\mathcal{D}}} \text{ is a subdeduction of } \underline{\underline{\Pi}}^i_j(\mathrm{Base}(\underline{\underline{\mathcal{D}}})) \text{ and } i,j \in N\}$$

$\overline{K}(\underline{\underline{\mathcal{D}}})$ is intended to represent the minimal cost of computing $\underline{\underline{\mathcal{D}}}$ from $\mathrm{Base}(\underline{\underline{\mathcal{D}}}) \cup \mathcal{E}$ using Ref with some pair (Δ_i, E_j). Given that $\{\Delta : i \in N\}$ contains each finite Γ-refinement, this intention is satisfied.

Assumption 1. $\{\Delta_i: i \in N\}$ contains each finite Γ-assignment.

Completeness concepts defined below will be relative to completeness of the underlying calculus Γ. Consequently it is convenient to assume that $\{\Delta_i: i \in N\}$ contains the maximal Γ-refinement consisting of all Γ-deductions:

Assumption 2. Δ_0 is the maximal Γ-refinement

B.2 Local Measures

The concepts defined below are concerned primarily with attributes of specific refinements and search strategies as opposed to global measures which relate these attributes to other refinements and search strategies. Local measures can be effectively evaluated by a combination of analytical and empirical techniques.

Refutable clause-sets (in $\mathcal{U}_{\mathcal{E}}$) for a refinement Δ_i are defined by

$$\mathcal{S}(i) =_{df} \{ \mathcal{C} \in \mathcal{U}_{\mathcal{E}} : \Delta_i \text{ contains a refutation of } \mathcal{C} \cup \mathcal{E} \} .$$

Thus, $\mathcal{S}(0) = \{ \mathcal{C} \in \mathcal{U}_{\mathcal{E}} : \mathcal{C}$ has a Γ-refutation$\}$ by Assumption 2.

Completeness of a refinement Δ_i may be defined probabilistically as the expectation that a Γ-refutable set $\mathcal{C}$ in $\mathcal{U}_{\mathcal{E}}$ has a Γ-refutation in Δ_i :

$$\text{Compl}(i) =_{df} \mu(\mathcal{S}(i)/(\mu(\mathcal{S}(0))$$

Notice that Compl(i) may be 1 even though Δ_i is not Γ-complete.

Relevancy (also known as efficiency ratio [49]) is an attribute of refutation procedures which compares the cost of the first solution found with the total cost of the computation:

$$\text{Rel}_j^i(\mathcal{C}) =_{df} \begin{cases} \overline{K}(\underline{\Pi}_j^i(\mathcal{C})(0))/K_j^i(\mathcal{C}) , & \text{if } \underline{\Pi}_j^i(\mathcal{C}) \text{ is a refutation;} \\ \overline{K}(\mathcal{C})/K_j^i(\mathcal{C}) , & \text{otherwise.} \end{cases}$$

If $\underline{\underline{\Pi}}^i_j(\mathcal{C})$ is a refutation then $\underline{\underline{\Pi}}^i_j(\mathcal{C})(0)$ is the first solution found. If $\underline{\underline{\Pi}}^i_j(\mathcal{C})$ is not a refutation then the "first solution" is to read $\mathcal{C}$ into the output queue and halt.

The expected relevancy $\int \mathrm{Rel}^i_j \, d\mu$ depends upon both the "irredundancy" of Δ_i and the "goal directedness" of E_j.

Power. Even if Δ_i is a strong and complete Γ-refinement, it does not follow that the expected cost of $\underline{\underline{\Pi}}^i_j$ will be less than the expected cost of $\underline{\underline{\Pi}}^0_j$, which uses the weakest Γ-refinement. The reason is that all the simplest refutations in $\Delta_0(\mathcal{C})$ may be filtered out by Δ_i, so that the refutation $\underline{\underline{\Pi}}^i_j(\mathcal{C})$ is quite costly to generate. Thus we define Power(i) on $\mathcal{I}(i)$ by

$$\mathrm{Power}(i)(\mathcal{C}) =_{df} \frac{\min\{\overline{K}(\underline{\underline{\mathcal{D}}}): \underline{\underline{\mathcal{D}}} \in \Delta_0(\mathcal{C} \cup \mathcal{E})\}}{\min\{\overline{K}(\underline{\underline{\mathcal{D}}}): \underline{\underline{\mathcal{D}}} \in \Delta_i(\mathcal{C} \cup \mathcal{E})\}}$$

The expected power $\int_{\mathcal{I}(i)} \mathrm{Power}(i) \, d\mu/\mu(\mathcal{I}(i))$ is a fair indication of how many "simple" refutations Δ_i filter out on its domain of completeness.

Convergence. Consider the problem-domain $\mathcal{U}^i_j$ defined by

$$\mathcal{U}^i_j =_{df} \{\mathcal{C} \in \mathcal{U}_{\mathcal{E}}: \mathcal{C} \text{ has a } \Gamma\text{-refutation } \underline{\underline{\mathcal{D}}} \text{ such that } \overline{K}(\underline{\underline{\mathcal{D}}}) \leq K^i_j(\mathcal{C})\}$$

$\mathcal{U}^i_j$ is the collection of clause-sets $\mathcal{C}$ having a Γ-refutation which is no more costly than the deduction $\underline{\underline{\Pi}}^i_j(\mathcal{C})$, regardless of whether the latter is a refutation or not. The convergence of Π^i_j on $\mathcal{U}_{\mathcal{E}}$ is given by

$$\mathrm{Conv}_j^i =_{df} \mu(\{\mathcal{C} \in \mathcal{U}_j^i : \Pi_j^i(\mathcal{C}) \text{ is a refutation}\})/\mu(\mathcal{U}_j^i)$$

Conv_j^i compares the expected completeness of Π_j^i with the expected completeness of an "ideal" proof procedure under the constraint that the "ideal" procedure never exerts more effort than Π_j^i on a given problem.

Convergence is dependent upon (expected) completeness, relevancy, and power. A search strategy which satisfies "Kowalski's maxim" [41],

> Search strategies should attempt to generate simpler before more complex proofs.

would appear more likely to yield highly convergent proof procedures than one which does not.

Suppose $\mathrm{Conv}_j^i = 1$. It follows that for "almost all" clause-sets $\mathcal{C} \in \mathcal{U}_\mathcal{E}$, if Π_j^i fails to find a refutation for $\mathcal{C}$ after a given amount c of computational effort, then there exists no Γ-refutation of $\mathcal{C}$ with complexity less than c. As $c \to \infty$, our conviction that $\mathcal{C}$ is consistent approaches certainty! For this reason, convergence is a far more useful performance measure for proof-procedures than is completeness, which alone tells us little or nothing about the relation between computational effort and probability of eventually finding a refutation.

B.3 Global Measures

A global performance measure compares the expected performance of a refutation procedure Π_j^i with the best expected performance of

any other procedure (in the space $\{\Pi^i_j: i,j \in N\}$). Consequently, global performance measures can be quite difficult to investigate empirically. The following paragraphs define several global measures pertaining to "efficiency" of proof procedures.

Search efficiency. Recall that Δ' is complete relative to Δ provided that if Δ contains a refutation for $\mathcal{C}$ then Δ' also contains a (possibly different) refutation for $\mathcal{C}$ ($\mathcal{C} \in \mathcal{U}_\mathcal{E}$). Search efficiency of $\underline{\Pi}^i_j$ compares the costs of $\underline{\Pi}^i_j$ with the costs of other proof procedures $\underline{\Pi}^{i'}_{j'}$ where Δ_i is complete relative to $\Delta_{i'}$:

$$SE^i_j =_{df} \min\{\int (K^{i'}_{j'}/K^i_j)d\mu: \Delta_{i'} \text{ is complete relative to } \Delta_i,\ i',j' \in N\}$$

Convergence efficiency simply compares convergence of Π^i_j with "best possible" convergence obtainable by relatively complete refinements:

$$CE^i_j =_{df} (\min\{Conv^{i'}_{j'}: \Delta_{i'} \text{ is complete relative to } \Delta_i, i',j' \in N\}/Conv^i_j).$$

Remark. While the concept of "best possible" proof procedure for a domain $\mathcal{U}_\mathcal{E}$ may be impossible to define empirically or even theoretically in general, there do exist restricted problem-domains such as sentence-recognition in context-free languages, for which near-optimal decision procedures have been developed. If $\mathcal{U}_{\mathcal{E}'}$ is such a domain, then it will usually be possible to represent a known

near-optimal procedure in the form Π_j^i for some Δ_i, E_j. Estimates of efficiency for a proof-procedure Π_j^i on possibly larger domains $\mathcal{U}_{\mathcal{E}}$ (where $\mathcal{E}' \subseteq \mathcal{E}$) can then be obtained empirically by comparing Π_j^i with $\Pi_{j'}^{i'}$ on $\mathcal{U}_{\mathcal{E}'}$. Some very interesting research along these lines has already been conducted.

C. THE DESIGN OF COMPLEXITY ORDERINGS

This appendix describes and illustrates two related methods of defining $\mathcal{E}$-complexity orderings (§2.3.5) for terms over a given vocabulary V. Each method has the same underlying motivation. We are given a finite axiom system $\mathcal{E}$ wherein the order of terms in equations is assumed to have some "intuitive significance". Specifically, it is intended that specialized proof procedures for $\models_{\mathcal{E}}$ will treat most or all of the equations $[s=t]$ in $\mathcal{E}$ as reduction rules whereby Rp-inferences

$$\{[s=t], B \vee q[\underline{r}]\} \vdash (B \vee q[t])\ \mathrm{mgu}\{r,s\}$$

are admitted but their symmetrical counterparts

$$\{[s=t], B \vee q[r]\} \vdash (B \vee q[s])\ \mathrm{mgu}\{r,t\}$$

are never admitted. Our objective is to approximate this "intuitive significance" while preserving completeness of our proof procedures.

This objective may be accomplished by using $\mathrm{ND}(\mathcal{E},\succ)$ (§2.3.8) as the resolution micro-refinement (§2.3.0) in a normal proof procedure for $\models_{\mathcal{E}}$ (§2.4.2), where $(\mathcal{E},\succ)$ is a reduction system (§2.3.4). In general, however, not all equations of $\mathcal{E}$ can be strictly ordered by $\succ$. For example, if $\mathcal{E}$ contains a "commutativity" axiom $[x \cdot y = y \cdot x]$ and $x \cdot y \succ y \cdot x$, then $y \cdot x \succ x \cdot y$ by invariance of $\succ$ under substitutions $([y/x, x/y])$. Moreover, presently available completeness results

for $ND(\mathcal{E},\succ)$ (§3) require that $\succ$ be an $\mathcal{E}'$-complexity ordering for some $\mathcal{E}' \subseteq \mathcal{E}$.

C.1 Introduction

How does the choice of the ordering relation $\succ$ affect the performance of a normal proof procedure based on $ND(\mathcal{E},\succ)$? Two performance-oriented features of $\succ$ might be described loosely as strength and direction.

Strength. $\succ_2$ is stronger than $\succ_1$ provided that $\{(s,t): s \succ_1 t \text{ or } t \succ_1 s\} \subseteq \{(s,t): s \succ_2 t \text{ or } t \succ_2 s\}$. For example, each "proper extension" of $\succ_1$ is stronger than $\succ_1$. In view of the role of $\succ$ in restricting the class of Rp-inferences admitted by $ND(\mathcal{E},\succ)$ we should make $\succ$ as strong as feasible without significantly increasing the computational cost of evaluating relations $(s \succ t)$ in a normal proof procedure.

Direction. Let us say that a reduction system $\underline{\mathcal{E}} = (\mathcal{E},\succ)$ converges to (a subset of) an $\mathcal{E}$-normal form $\mathcal{N}$ provided that if $\Delta = ND(\mathcal{E},\succ)$ then $NF(Cl(\mathcal{E}),\succ) \subseteq \mathcal{N}$ (for each fair enqueuing function Enq). If $\underline{\mathcal{E}}$ converges to $\mathcal{N}$ then a normal proof procedure based on $ND(\mathcal{E},\succ)$ will eventually begin reducing terms to members of $\mathcal{N}$ before unifying them in Rp and Cut inferences. $\mathcal{N}$ may be thought of as a "direction" for $\underline{\mathcal{E}}$ in a partially ordered space of normal form representation.

Efficient representations. Given reduction systems $\underline{\mathcal{E}}_1 = (\mathcal{E},\succ_1)$ and $\underline{\mathcal{E}}_2 = (\mathcal{E},\succ_2)$, we can compare $\succ_1$ and $\succ_2$ in terms of $\mathcal{E}$-normal forms to which $\underline{\mathcal{E}}_i$ converges. Given that $\underline{\mathcal{E}}_i$ converges to $\succ_i$, we

may be motivated to devise proof procedures for $\models_{\mathcal{E}}$ which represent and process terms of $\mathcal{M}_i$ with optimal efficiency. Thus, we may prefer $\succ_1$ to $\succ_2$ on grounds that normal proof procedures for $\models_{\mathcal{E}}$ can be made to operate more efficiently on (the representation for terms in) $\mathcal{M}_1$ than on $\mathcal{M}_2$. (See §D.0 (Remark).)

Two related approaches to the design of $\mathcal{E}$-complexity orderings are investigated below. The first approach (§C.2) uses a weighting function $w: V_F \to N$ to define a complexity order $\succ_w$ (§2.3.5) wherein the "complexity" of a constant term t is the sum of weights of initial operators of subterms of t. This approach is illustrated by an application to group theory described by Knuth and Bendix [39]. Beginning with a reduction system $\underline{G} = (G, \succ_w)$ where G consists of three standard axioms for group theory, we use a normal proof procedure based on $ND(G, \succ_w)$ to derive a ten-axiom system R such that $(R, \succ_w)$ is canonical (§2.3.4) and closed (§3.2.3).

The second approach (§C.3) defines an $\mathcal{E}$-complexity ordering $\succ_{\underline{\mathcal{E}}}$ by means of an $\mathcal{E}$-normal mapping ν on $\mathcal{T}_V$ (§1.2.9) and a function $\phi: \mathcal{T}_V \to N$ which measures the "cost" of computing $\nu(\overline{u})$ from u. Given terms u,v such that $\nu(u) = \nu(v)$, we say that $u \succ_{\underline{\mathcal{E}}} v$ provided that either $\phi(u) > \phi(v)$ or else $\phi(u) = \phi(v)$ and $u \succ_w v$, where $\succ_w$ is some "weighting function" complexity ordering. If $\nu(u\theta) \neq \nu(v\theta)$ for all θ then we say that $u \succ_{\underline{\mathcal{E}}} v$ provided that $\nu(u) \succ_w \nu(v)$.

The second approach is illustrated (§C.5) by a D-complexity ordering $\succ_{\underline{D}}$ for commutative rings, integral domains, and fields,

where D is the canonical system {D1,D2,D3,D4} :

D1: $[(x+y)+z = x+(y+z)]$;

D2: $[(x\cdot y)\cdot z = x\cdot(y\cdot z)]$;

D3: $[(x+y)\cdot z = x\cdot z + y\cdot z]$;

D4: $[z\cdot(x+y) = x\cdot z + y\cdot z]$.

It is easily verified that $(D,\succ_{\underline{D}})$ converges to NF(D), and that $(D,\succ_w)$ does not converge to NF(D) for any "weighting function" complexity ordering $\succ_w$. (See §D.0, Remark.)

The analysis of $\succ_D$ is facilitated by an effective characterization of completeness for total determinative reduction systems in §C.4.

C.2 Complexity Orderings Based on Weighting Functions

Complexity orderings $\succ_w$ based on a weighting function $w: V_F \to N$ are defined in §2.3.5. These orderings yield very useful refinements $(ND(\mathcal{E},\succ_w))$ for applications in group theory and other equational systems. The following application in group theory illustrates how the procedure $\underline{C\ell}$ (§3.2.3) may be used as a proof procedure for equational systems.

The vocabulary V_E for group theory contains three operation constants $(\cdot,^-,1)$; $V_R = 0$.

Axioms. G = {G1,G2,G3} :

G1: $[(x\cdot y)\cdot z = x\cdot(y\cdot z)]$; (Associativity)

G2: $[1\cdot x = x]$; (Left identity)

G3: $[x^-\cdot x = 1]$. (Left inverse)

Complexity ordering. Define w by

$w(1) = 1$;

$w(^-) = 0$;

$w(\cdot) = 0$.

Then w is a weighting function for V , provided that - is taken to be the last element of V_F in lexical order. Define the complexity ordering $>_w$ for $\mathcal{T}_V$ as in §2.3.5. Observe that each equation of $\mathcal{E}$ is a reduction according to $\succ_w$.

The following deduction from G is complete in $ND(\mathcal{E},\succ_w)^-$ (§1.3.4). It contains 101 clauses. Of these, 91 are deleted either before or immediately after the last equation is generated, because they reduce to identities in the context of the remainder (R):

R0: $[1^- = 1]$
R1: $[x^{--} = x]$
R2: $[1 \cdot x = x]$
R3: $[x \cdot 1 = x]$
R4: $[x^- \cdot x = 1]$
R5: $[x \cdot x^- = 1]$
R6: $[(x \cdot y)^- = y^- \cdot x^-]$
R7: $[x \cdot (x^- \cdot z) = z]$
R8: $[x^- \cdot (x \cdot z) = z]$
R9: $[(x \cdot y) \cdot z = x \cdot (y \cdot z)]$

Each of these is a reduction according to $\succ_w$, and each is irreducible with respect to the others. Indeed, the reduction system $(R,\succ_w)$ is closed, as noted (in effect) by Knuth and Bendix [39].

Format of proof. Each line has the format

(In Clause Source Out Disposition)

where

In = order of entry of Clause in Q ;

Clause = Axiom or conclusion of an inference;

Source = Ax(iom) or inference rule used:

Cut(m,n)-pairwise resolution (on clauses m,n),

Rp(m,n)- replacement (from clause m into clause n),

CNR(m)- complete normal reduction from clause m based on currently active equations;

Out = order of exit from Q ;

Disp(osition): if Clause is deleted, how (why?)

CNR(*) - Clause reduces to true (e.g., an identity) in current equations;

CNR (+) - Clause reduces to useful result; and

C - Clause reduces to true following addition of last clause (97 below).

When a clause is selected from Q , its entry number is prefixed by "-" and its exit feature is assigned the next higher exit number. When this clause has been "resolved" with all clauses currently prefixed by "+", its "-" is converted to "+". When a clause reduces to true, or is subsumed by the current output clause, its prefix is converted to "*" . Output clauses (R) are those prefixed by + or * .

Fig. 1. A Group Theory Derivation

In	Clause		Out	Disposition
+1.	$1 \cdot x = x$	Ax	1	
+2.	$x^{-} \cdot x = 1$	Ax	2	
+3.	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	Ax	3	
*4.	$y \cdot z = 1 \cdot (y \cdot z)$	Rp(1,3)	4	4 CNR(*)
*5.	$z = x^{-} \cdot (x \cdot z)$	Rp(2,3)	5	
*6.	$(w \cdot (x \cdot y)) \cdot z = (w \cdot x) \cdot (y \cdot z)$	Rp(3,3)	6	6 CNR(*)
+7.	$x^{-} \cdot (x \cdot z) = z$	Rp(5,[x=x])	5	
*8.	$1^{-} \cdot z = z$	Rp(1,7)	8	23 CNR(*)
*9.	$x^{--} \cdot 1 = x$	Rp(2,7)	7	21 CNR(+)
±10.	$(x \cdot y)^{-} (x \cdot (y \cdot z)) = z$	Rp(3,7)	15	C
*11.	$y \cdot z = w^{-} \cdot ((w \cdot y) \cdot z)$	Rp(7,3)		
*12.	$w \cdot z = w^{--} \cdot (1 \cdot z)$	Rp(9,3)	9	9 CNR(+)
*13.	$z^{---} \cdot z = 1$	Rp(9,7)		28 CNR(*)
*14.	$1 = 1$	Rp(8,2)		0
*15.	$y \cdot z = 1^{-} \cdot (y \cdot z)$	Rp(8,3)		23 CNR(*)
*16.	$w \cdot z = w^{--} \cdot z$	CNR(12)		9
*17.	$w^{--} \cdot z = w \cdot z$	Rp(16),[x=x])	9	28 CNR(*)
+18.	$w \cdot w^{-} = 1$	Rp(17,2)	13	
*19.	$(x \cdot y) \cdot z = x^{--} \cdot (y \cdot z)$	Rp(17,3)		28 CNR(*)
+20.	$x \cdot (x^{-} \cdot z) = z$	Rp(17,7)	14	
+21.	$x \cdot 1 = x$	Rp(17,9)	10	
*22.	$1 = 1$	Rp(21,1)		22 CNR(*)
+23.	$1^{-} = 1$	Rp(21,2)	11	
*24.	$(x \cdot y) = x(y \cdot 1)$	Rp(21,3)		21 CNR(*)
*25.	$x \cdot z - x \cdot (1 \cdot z)$	Rp(21,3)		1 CNR(*)
*26.	$x^{-} \cdot x = 1$	Rp(21,7)		2 CNR(*)
*27.	$1^{-} = 1$	Rp(21,8)		23 CNR(*)
+28.	$x^{--} = x$	Rp(21,9)	12	
*29.	$w^{--} = w \cdot 1$	Rp(21,17)		28 CNR(*)
*30.	$1 \cdot 1 = 1$	Rp(23,2)		1 CNR(*)

In	Clause	Source		Disposition
*31.	$1 \cdot (1 \cdot z) = z$	Rp(23,7)	1	CNR(*)
*32.	$1^{-} \cdot z = 1 \cdot z$	Rp(23,17)	23	CNR(*)
*33.	$x \cdot x^{-} = 1$	Rp(28,2)	18	CNR(*)
*34.	$x \cdot (x^{-} \cdot z) = z$	Rp(28,7)	20	CNR(*)
*35.	$1^{-} = 1$	Rp(1,18)	23	CNR(*)
*36.	$1 \cdot z = x \cdot (x^{-} \cdot z)$	Rp(18,3)	20	CNR(*)
37.	$x \cdot (y \cdot (x \cdot y)^{-}) = 1$	Rp(3,18)		C
*38.	$x^{-} \cdot 1 = x^{-}$	Rp(18,7)	21	CNR(*)
*39.	$1 \cdot 1 = 1$	Rp(23,18)		CNR(*)
*40.	$x^{-} \cdot x = 1$	Rp(28,18)		CNR(*)
*41.	$1^{-} \cdot z = z$	Rp(1,20)		CNR(*)
*42.	$x \cdot 1 = x$	Rp(2,20)		CNR(*)
43.	$x \cdot (y \cdot ((x \cdot y)^{-} \cdot z)) = z$	Rp(3,20)		C
*44.	$x \cdot 1 = x^{--}$	Rp(18,20)		CNR(*)
*45.	$x \cdot x^{-} = 1$	Rp(21,20)		CNR(*)
*46.	$1 \cdot (1 \cdot z) = z$	Rp(23,20)		CNR(*)
*47.	$x^{-}(x \cdot z) = z$	Rp(28,20)		CNR(*)
*48.	$y^{-} \cdot (1 \cdot (y \cdot z)) = z$	Rp(1,10)		CNR(*)
*49.	$y^{-} \cdot (y \cdot z) = z$	Rp(1,10)		CNR(*)
*50.	$x^{-}(x \cdot z) = z$	Rp(1,10)		CNR(*)
*51.	$1^{-} \cdot (x^{-} \cdot (x \cdot z))$	Rp(2,10)		CNR(*)
52.	$((y \cdot z)^{-} \cdot y)^{-} \cdot 1 = z$	Rp(2,10)		C
53.	$(x . z^{-})^{-} \cdot (x \cdot 1) = z$	Rp(2,10)		C
54.	$(w \cdot (x \cdot y))^{-} \cdot (w \cdot x) \cdot (y \cdot z)) = z$	Rp(3,10)		C
55.	$((w \cdot x) \cdot y)^{-} \cdot (w \cdot (x \cdot (y \cdot z)) = z$	Rp(3,10)		C
56.	$(x \cdot (w \cdot y))^{-} \cdot (x \cdot (w \cdot (y \cdot z))) = z$	Rp(3,10)		C
57.	$y^{-} \cdot (w^{-} \cdot ((w \cdot y) \cdot z)) = z$	Rp(7,10)		CNR(*)
*58.	$(x^{-} \cdot x)^{-} \cdot z = z$	Rp(7,10)		CNR(*)
59.	$(x \cdot w^{-})^{-} \cdot (x \cdot z) = w \cdot z$	Rp(7,10)		C
*60.	$1^{-} \cdot (w \cdot (w^{-} \cdot z)) = z$	Rp(18,10)		CNR(*)
61.	$(x \cdot y)^{-} \cdot (x \cdot 1) = z$	Rp(18,10)		C
*62.	$w^{-} \cdot (x \cdot ((x^{-} \cdot w) \cdot z)) = z$	Rp(20,10)		CNR(*)

In	Clause	Source		Disposition
*63.	$(x \cdot x^-)^- \cdot z = z$	Rp(20,10)		CNR(*)
+64.	$(x \cdot y)^- \cdot (x \cdot z) = y^- z$	Rp(20,10)	16	C
*65.	$x^- \cdot (x \cdot (1 \cdot z)) = z$	Rp(21,10)		CNR(*)
66.	$(x \cdot y)^- \cdot (x \cdot y) = 1$	Rp(21,10)		C
*67.	$y^- \cdot (1 \cdot z) = y^- \cdot z$	Rp(1,64)		CNR(*)
*68.	$1^- \cdot (x^- \cdot z) = x^- \cdot z$	Rp(2,64)		CNR(*)
69.	$(x \cdot (y \cdot w))^- \cdot ((x \cdot y) \cdot z) = w^- \cdot z$	Rp(3,64)		C
70.	$((x \cdot w) \cdot y)^- \cdot (x \cdot (w \cdot z)) = y^- \cdot z$	Rp(3,64)		C
71.	$w^- \cdot (x^- \cdot z) = (x \cdot w)^- \cdot z$	Rp(7,64)		C
72.	$(x^- \cdot y)^- \cdot z = y^- \cdot (x \cdot z)$	Rp(7,64)		CNR(*)
73.	$w = y^- \cdot (y \cdot w)$	Rp(10,64)		CNR(*)
74.	$w^- \cdot ((x \cdot y)^- \cdot z) = (x \cdot (y \cdot w))^- \cdot z$	Rp(10,64)		C
75.	$((x \cdot y)^- \cdot w)^- \cdot z = w^- \cdot (x \cdot (y \cdot z))$	Rp(10,64)		C
76.	$(y^- \cdot w)^- \cdot ((x \cdot y)^- \cdot ((x \cdot w) \cdot z) = z$	Rp(64,10)		C
77.	$((w \cdot y) \cdot w)^- \cdot (y^- \cdot z) = z$	Rp(64,10)		C
78.	$(x \cdot (w \cdot y))^- \cdot (x \cdot (y^- \cdot z)) = w \cdot z$	Rp(64,10)		C
*79.	$1^- \cdot (x \cdot z) = x^{--} \cdot z$	Rp(18,64)		CNR(*)
+80.	$(x \cdot y)^- \cdot 1 = y^- \cdot x^-$	Rp(18,64)	17	CNR(+)
81.	$y^- \cdot (w \cdot z) = (w^- \cdot y)^- \cdot z$	Rp(20,64)		C
82.	$(x \cdot y)^- \cdot z = y^- \cdot (x^- \cdot z^-)$	Rp(20,64)		C
83.	$x^- \cdot (x \cdot z) = 1^- \cdot z$	Rp(21,64)		CNR(*)
84.	$(x \cdot y)^- \cdot x = y^- \cdot 1$	Rp(21,64)		C
85.	$y^- \cdot 1 = y^- \cdot 1^-$	Rp(1,80)		CNR(*)
86.	$1^- \cdot 1 = x^- \cdot x^{--}$	Rp(2.80)		CNR(*)
87.	$(x \cdot (y \cdot z))^- \cdot 1 = z^- \cdot (x \cdot y)^-$	Rp(3,80)		C
88.	$(y^- \cdot x^-) \cdot z = (x \cdot y)^- \cdot (1 \cdot z)$	Rp(80,3)		C
89.	$z^- \cdot 1 = (x \cdot z)^- \cdot x^{--}$	Rp(7,80)		C
90.	$(x \cdot y)^- \cdot (y^- \cdot x^-) = 1$	Rp(80,7)		C
91.	$(y^- \cdot x^-)^- \cdot ((x \cdot y)^- \cdot (1 \cdot z)) = z$	Rp(80,10)		C
92.	$(w \cdot (x \cdot y^-))^- \cdot (w \cdot ((y^- \cdot x^-) \cdot 1)) = 1$	Rp(80,10)		C
93.	$z^- \cdot 1 = (x \cdot (y \cdot z))^- \cdot (x \cdot y)^{--}$	Rp(10,80)		C

In	Clause	Source		
94.	$1^{-}\cdot 1 = x^{--}\cdot x^{-}$	Rp(18,80)		CNR(*)
95.	$(x\cdot y)\cdot(y^{-}\cdot x^{-}) = 1$	Rp(80,20)		CNR(*)
96.	$z^{-}\cdot 1 = (x^{-}\cdot z)^{-}\cdot x^{-}$	Rp(20,80)		C
-97.	$(x\cdot y)^{-} = y^{-}\cdot x^{-}$	Rp(21,80)	18	
98.	$x^{-}\cdot 1 = 1^{-}\cdot x^{-}$	Rp(21,80)		CNR(*)
99.	$(y^{-}\cdot z)^{-}\cdot 1 = (x\cdot z)^{--}\cdot(x\cdot y)^{-}$	Rp(64,80)		C
100.	$(y^{-}\cdot x^{-})^{-}\cdot((x\cdot y)^{-}\cdot z) = 1^{-}\cdot z$	Rp(80,64)		C
101.	$((x\cdot y)^{-}\cdot y)^{-}\cdot(y^{-}\cdot x^{-}) = y^{-}\cdot 1$	Rp(80,64)		C

C.3 $\mathcal{E}$-Complexity Orderings Based on an $\mathcal{E}$-Canonical Mapping

Let $\succ_W$ be a weighting-function complexity ordering as defined in §2.3.5, and suppose that $\mathcal{E}$ is finitary and complete--i.e.,

(i) $\mathcal{E}$ is finite;

(ii) $\rightarrow_{\mathcal{E}}^*$ is a partial ordering which satisfies D.C.C; and

(iii) if $u =_{\mathcal{E}} v$ then $u \rightarrow_{\mathcal{E}}^* t$ and $v \rightarrow_{\mathcal{E}}^* t$ for some term t.

It follows by Proposition 10 in §2.3.3 that $NF(\mathcal{E}, \rightarrow_{\mathcal{E}}^*)$ is an $\mathcal{E}$-canonical form. Indeed, for each term u there exists a term $v \in NF(\mathcal{E})$ such that $u \Rightarrow_{\mathcal{E}}^* v$, because of (ii). Moreover, v is unique (because of (iii): Choose $v_1, v_2 \in NF(\mathcal{E})$ such that $u \Rightarrow_{\mathcal{E}}^* v_i$ $(i=1,2)$. Then $v_1 =_{\mathcal{E}} v_2$, whence $v_i \rightarrow_{\mathcal{E}}^* v$ for some v. Therefore $v_1 = v = v_2$).

$$\nu(u) =_{df} \text{(the unique term } v \in NF(\mathcal{E}) \text{ such that } u \Rightarrow_{\mathcal{E}}^* v\text{)}.$$

The purpose of this section is to define from $(\mathcal{E}, \succ_W)$ a class of $\mathcal{E}$-complexity orderings $\succ_{\mathcal{E}}$ wherein each equation of $\mathcal{E}$ is a "reduction":

(iv) if $[s=t] \in \mathcal{E}$ then $s \succ_{\underline{\mathcal{E}}} t$.

It follows that

(v) $u \succeq_{\underline{\mathcal{E}}} \nu(u)$ $(u \in \mathcal{T}_V)$; and

(vi) $NF(\mathcal{E}, \succ_{\underline{\mathcal{E}}}) \subseteq NF(\mathcal{E})$.

$\succ_{\underline{\mathcal{E}}}$ is actually defined so that

(vii) $\nu(s) \succ_{\underline{\mathcal{E}}} \nu(t)$ iff $\nu(s) \succ_W \nu(t)$; and

(viii) if $s \succ_{\underline{\mathcal{E}}} t$ then $\nu(t) \not\succ_W \nu(s)$.

In order to accomplish this purpose, let $\underline{\mathcal{E}} = (\mathcal{E}, \succ_W, \phi, Q)$ where

(ix) ϕ: $\mathcal{J}_V \to N$ is computable (a "cost function" for ν); and

(x) Q: $\mathcal{J}_V \times \mathcal{J}_V \to \{0,1\}$ is computable.

Define $\succ_{\underline{\mathcal{E}}}$ by

$$(s \succ_{\underline{\mathcal{E}}} t) =_{df} [\underline{\text{If}}\ \nu(s) = \nu(t)$$
$$\underline{\text{then}}\ \underline{\text{if}}\ \phi(s) = \phi(t)$$
$$\underline{\text{then}}\ (s \succ_W t)$$
$$\underline{\text{else}}\ (\phi(s) \succ \phi(t))$$
$$\underline{\text{else}}\ \underline{\text{if}}\ Q(s,t) = 0$$
$$\underline{\text{then}}\ \nu(s) >_W \nu(t)$$
$$\underline{\text{else}}\ \underline{\text{false}}].$$

A straightforward attempt to verify that $\succ_{\underline{\mathcal{E}}}$ is an $\underline{\mathcal{E}}$-complexity ordering (below) generates the following set of sufficient conditions:

(C1) $\nu(u[t]) \succeq_W \nu(t)$.

(C2) If $\nu(s) \succ_W \nu(t)$
then $\nu(u[s]\theta) \succ_W \nu(u[t]\theta)$ $(\theta \in \Sigma_{\mathcal{E}})$.

(C3) If $[s=t] \in \mathcal{E}$ and $\phi(s) \neq \phi(t)$ then $\phi(s) = \phi(t)$ and $s \succ_W t$.

(C4) $\phi(u[t]) \succeq \phi(t)$.

(C5) If $\nu(s) = \nu(t)$
then $(\phi(u[s]\theta) > \phi(u[t]\theta)) = (\phi(s) > \phi(t))$ $(\theta \in \Sigma_{\mathcal{E}})$.

(C6) If $Q(s,t) = 0$ then $\nu(s\theta) \neq \nu(t\theta)$ $(\theta \in \Sigma_{\mathcal{E}})$.

(C7) If $Q(s,t) = 0$ then $Q(u[s]\theta, u[t]\theta) = 0$ $(\theta \in \Sigma_{\mathcal{E}})$.

(C8) If $\nu(s) \succ_W \nu(t)$ or $\nu(t) \succ_W \nu(s)$ then $Q(s,t) = 0$.

Proposition 1. Suppose $\mathcal{E}$ satisfies (C1)-(C8). Then $\succ_{\mathcal{E}}$ is an $\mathcal{E}$-complexity ordering satisfying (iv)-(viii).

Proof. We verify each condition in the definition of $\mathcal{E}$-complexity ordering in sequence (§2.3.5).

$\succ_{\mathcal{E}}$ is decidable because each relation in the explicit definition of $\succ_{\mathcal{E}}$ is decidable. (We are assuming Q to be computable, whence the relation $(Q(s,t) = 0)$ is decidable.)

$\succ_{\mathcal{E}}$ is monotone. This we demonstrate by a subcase analysis:

S: $s \succ_{\mathcal{E}} t$ (Premise).

S1: $\nu(s) = \nu(t)$, whence $\nu(u[s]) = \nu(u[t])$ by definition of ν.

S1.1: $\phi(s) = \phi(t)$, whence $s \succ_W t$, and $u[s] \succ_W u[t]$ by monotonicity of $\succ_W$. Now $[\phi(u[s]) > \phi(u[t])]$ $= [\phi(s) > \phi(t)] =$ false, and $[\phi(u[t]) > \phi(u[s])]$ $= [\phi(t) > \phi(s)] =$ false by (C5), whence $\phi(u[s]) = \phi(u[t])$.

It follows that $u[s] \succ_{\mathcal{E}} u[t]$.

S1.2: $\phi(s) \neq \phi(t)$, whence $\phi(s) \succ \phi(t)$ by (S). It follows by (C5) that $\phi(u[s]) > \phi(u[t])$, whence $u[s] \succ_{\mathcal{E}} u[t]$.

S2: $\nu(s) \neq \nu(t)$, whence $\nu(s) \succ_W \nu(t)$ by (S).

It follows by (C2) that $\nu(u[s]) \succ_W \nu(u[t])$, whence $u[s] \succ_{\mathcal{E}} u[t]$ by (C8).

Thus, (S) $\models u[s] \succ_{\mathcal{E}} u[t]$.

$\succ_{\underline{\varepsilon}}$ is a partial ordering. It is easily verified that $\succ_{\underline{\varepsilon}}$ is anti-symmetric (and anti-reflexive). It suffices to show that $\succ_{\underline{\varepsilon}}$ is transitive by a subcase analysis:

S: $s \succ_{\underline{\varepsilon}} t$ and $t \succ_{\underline{\varepsilon}} u$ (Premise)

S1: $\nu(s) = \nu(t)$.

S1.1: $\phi(s) = \phi(t)$, whence $s \succ_W t$.

S1.1.1: $\nu(t) = \nu(u)$.

S1.1.1.1: $\phi(t) = \phi(u)$. Then $s \succ_W u$ because $\succ_W$ is transitive, whence $s \succ_{\underline{\varepsilon}} u$.

S1.1.1.2: $\phi(t) \neq \phi(u)$. Then $\phi(t) > \phi(u)$, whence $\phi(s) > \phi(u)$ by (S1.1). It follows that $s \succ_{\underline{\varepsilon}} u$ because $\nu(s) = \nu(u)$ by (S1.1.1), (S1).

S1.1.2: $\nu(t) \neq \nu(u)$, whence $Q(t,u) = 0$ and $\nu(t) >_W \nu(u)$ by $(t \succ_{\underline{\varepsilon}} u)$. Therefore $\nu(s) \succ_W \nu(u)$ by (S1) and $Q(s,u) = 0$ by (C8). It follows that $s \succ_{\underline{\varepsilon}} u$.

S1.2: $\phi(s) \neq \phi(t)$, whence $\phi(s) > \phi(t)$.

S1.2.1: $\nu(t) = \nu(u)$, whence $\nu(s) = \nu(u)$ by (S1).

S1.2.1.1: $\phi(t) = \phi(u)$, whence $\phi(s) > \phi(u)$ by (S1.2). It follows that $s \succ_{\underline{\varepsilon}} u$.

S1.2.1.2: $\phi(t) \neq \phi(u)$, whence $\phi(t) > \phi(u)$ (by (S)). Therefore $\phi(s) > \phi(t)$ by (S1.2), whence $s \succ_{\underline{\varepsilon}} u$.

S1.2.2: $\nu(t) \neq \nu(u)$, whence $Q(t,u) = 0$ and $\nu(t) \succ_W \nu(u)$. Therefore $Q(s,u) = 0$ and $\nu(s) \succ_W \nu(u)$ by (C8) and (S1) whence $s \succ_{\underline{\varepsilon}} u$.

S2: $\nu(s) \neq \nu(t)$, whence $Q(s,t) = 0$ and $\nu(s) \succ_W \nu(t)$ by (S).

S2.1: $\nu(t) = \nu(u)$, whence $\nu(s) \succ_W \nu(u)$ by (S2) and $Q(s,u) = 0$ by (C8). Therefore $s \succ_{\underline{\underline{\varepsilon}}} u$.

S2.2: $\nu(t) \neq \nu(u)$, whence $\nu(t) \succ_W \nu(u)$ by (S) and $\nu(s) \succ_W \nu(u)$ by (S2). Therefore $Q(s,u) = 0$ and $s \succ_{\underline{\underline{\varepsilon}}} u$ by (C8).

(S) $\models (s \succ_{\underline{\underline{\varepsilon}}} u)$.

$\succ_{\underline{\underline{\varepsilon}}}$ well-orders constant terms of $\mathcal{T}_V$. It suffices to show that either $(s \succ_{\underline{\underline{\varepsilon}}} t)$ or $(t \succ_{\underline{\underline{\varepsilon}}} s)$ for each pair s,t of constant terms (trivial, by (C8) and definition of $\succ_W$) and that $\succ_{\underline{\underline{\varepsilon}}}$ satisfies D.C.C. (§2.3.2). Let $[t]_\nu =_{df} \{s: \nu(s) = \nu(t)\}$. Clearly $\succ_{\underline{\underline{\varepsilon}}}$ well-orders the constant terms of $[t]_\nu$, by the fact that $\succ_W$ is a complexity ordering. It is equally clear from (C8) that $\succ_{\underline{\underline{\varepsilon}}}$ well-orders the constant terms of $\{\nu(t): t \in \mathcal{T}_V\}$. D.C.C. follows from the observation that if $\nu(s) \succ_W \nu(t)$ where s and t are constant terms, then $u \succ_{\underline{\underline{\varepsilon}}} v$ for all pairs (u,v) of constant terms in $[s]_\nu \times [t]_\nu$.

$\underline{u[t] \succeq_{\underline{\underline{\varepsilon}}} t}$. Again we consider subcases:

S1: $\nu(u[t]) = \nu(t)$.

S1.1: $\phi(u[t]) = \phi(t)$. Then $(u[t] \succeq_{\underline{\underline{\varepsilon}}} t) = (u[t] \succeq_W t) = \underline{\text{true}}$.

S1.2: $\phi(u[t]) \neq \phi(t)$. Then $\phi(u[t]) > \phi(t)$ by (C4), whence $u[t] \succ_{\underline{\underline{\varepsilon}}} t$.

S2: $\nu(u[t]) \neq \nu(t)$. Then $\nu(u[t]) \succ_W \nu(t)$ by (C1), whence $Q(u[t],t) = 0$ by (C8).

$\models (u[t] \succeq_{\underline{\underline{\varepsilon}}} t)$.

If $s \succ_{\underline{\underline{\mathcal{E}}}} t$ then $s\theta \succ_{\underline{\underline{\mathcal{E}}}} t\theta$ $(\theta \in \Sigma_{\mathcal{E}})$:

S: $s \succ_{\underline{\underline{\mathcal{E}}}} t$ and $\theta \in \Sigma_{\mathcal{E}}$.

S1: $\nu(s) = \nu(t)$, whence $\nu(s\theta) = \nu(t\theta)$ by definition of ν .

S1.1: $\phi(s) = \phi(t)$, whence $s \succ_W t$ by (S), and $s\theta \succ_W t\theta$ because $\succ_W$ is a complexity ordering. Therefore $s\theta \succ_{\underline{\underline{\mathcal{E}}}} t\theta$ by (S1).

S1.2: $\phi(s) \neq \phi(t)$, whence $\phi(s) > \phi(t)$ by (S). Therefore $\phi(s\theta) > \phi(t\theta)$ by (C5). It follows by (S1) that $s\theta \succ_{\underline{\underline{\mathcal{E}}}} t\theta$.

S2: $\nu(s) \neq \nu(t)$, whence $\nu(s) \succ_W \nu(t)$ by (S), and $\nu(s\theta) \succ_W \nu(t\theta)$ by (C2). It follows by (C8) that $s\theta \succ_{\underline{\underline{\mathcal{E}}}} t\theta$.

(S) $\models (s\theta \succ_{\underline{\underline{\mathcal{E}}}} t\theta)$.

It follows by $(NF(\mathcal{E}) \neq 0)$ that $\succ_{\underline{\underline{\mathcal{E}}}}$ is an $\mathcal{E}$-complexity ordering. Conditions (iv)-(viii) above are easily verified. ((iv) follows from (C3).)∎

C.4 Canonical Reduction Systems

Suppose that $\underline{\underline{\mathcal{E}}} = (\mathcal{E}, \succ)$ where $\mathcal{E}$ and $\succ$ satisfy (i)-(iv):

(i) $\mathcal{E}$ is a finite set of equations $[s=t]$ where each variable in t is also in s .

(ii) $\succeq$ is an invariant partial ordering on $\mathcal{T}_V$.

(iii) Every descending chain in $\succeq$ is finite.

(iv) $s \succ t$ for each equation $[s=t]$ in $\mathcal{E}$.

In other words, let $\underline{\underline{\mathcal{E}}}$ be an equational, determinative, total, and

finitary reduction system (§2.3.4).

For the case where $\succ$ is a complexity ordering $\succ_W$, Knuth and Bendix prove the following [40, Theorem 5 (Corollary)]:

Proposition 2. Suppose that (1) holds for each equation [u=v] in $\mathcal{E}$:

If $\mathcal{E}^{\sim}$ contains [s=t], sharing no variables with [u=v], and $u = u'[r]$ where $r \notin V_I$ and $mgu\{r,s\} = \theta \in \Sigma_V$, then

$$u[t]\theta \to^*_{\mathcal{E}} v' \quad \text{and} \quad v\theta \to^*_{\mathcal{E}} v' \quad \text{for some term } v'. \tag{1}$$

Then $\hat{\mathcal{E}}$ is finitary, complete, and canonical.

Corollary. Suppose that (1) holds for each equation [u=v] in $\mathcal{E}$. Then $\mathcal{E}$ is normally complete.

Proof. If $u \to^*_{\mathcal{E}} v \in NF(\mathcal{E},\succ)$ then $u \Rightarrow^*_{\mathcal{E}} v$ because $\mathcal{E}$ is canonical.

We show below that Proposition 2 holds for any reduction system $\mathcal{E}$ satisfying (i)-(iv).

An $\mathcal{E}$-realization (for $u =_{\mathcal{E}} v$)) is a list $(u_0,\cdots,u_n)$ such that $u_0 = u$, $u_n = v$, and either $u_k \to_{\mathcal{E}} u_{k+1}$ or $u_{k+1} \to_{\mathcal{E}} u_k$ for $k=0,\cdots,n-1$.

Define $\sim_{\mathcal{E}}$ and $\delta_{\mathcal{E}}$ on $\mathcal{T}_V$ by

$(u \sim_{\mathcal{E}} v) =_{df}$ (there exists an $\mathcal{E}$-realization for $(u =_{\mathcal{E}} v)$.

$$\delta_{\mathcal{E}}(u,v) =_{df} \begin{cases} \min\{n: \text{there exists an } \mathcal{E}\text{-realization } (u_0,\cdots,u_n) \\ \qquad \text{for } (u =_{\mathcal{E}} v)\}, & \text{if } u \sim_{\mathcal{E}} v; \\ \infty, & \text{otherwise.} \end{cases}$$

Lemma 3. $u \sim_{\underline{\mathcal{E}}} v$ iff $u =_{\mathcal{E}} v$.

Proof. Clearly $(u \sim_{\underline{\mathcal{E}}} v)$ implies that $(u =_{\mathcal{E}} v)$. Moreover, $\sim_{\underline{\mathcal{E}}}$ is a monotone equivalence relation on $\mathcal{T}_V$. It follows by (i) and (ii) that $\sim_{\underline{\mathcal{E}}}$ is substutitive. Thus, $\sim_{\underline{\mathcal{E}}}$ is an equality relation on $\mathcal{T}_V$. Finally, (iv) implies that $s \sim_{\underline{\mathcal{E}}} t$ for each equation $[s=t]$ in $\mathcal{E}$. Since $=_{\mathcal{E}}$ is the smallest equality relation on $\mathcal{T}_V$ such that $s =_{\mathcal{E}} t$ for each equation $[s=t]$ in $\mathcal{E}$, it follows that $(u =_{\underline{\mathcal{E}}} v)$ implies $(u \sim_{\underline{\mathcal{E}}} v)$.∎

Now Proposition 3 in §2.3.4 shows that if $\underline{\mathcal{E}}$ is complete then $\underline{\mathcal{E}}$ is canonical. Thus, it suffices to show that $\underline{\mathcal{E}}$ is complete. The following lemma generalizes Theorem 4 in [39] to reduction systems $\underline{\mathcal{E}}$ satisfying (i)-(iv):

Lemma 4. $\underline{\mathcal{E}}$ is complete iff the following "lattice condition" is satisfied:

$$\text{If } u \to_{\underline{\mathcal{E}}} u_1 \text{ and } u \to_{\underline{\mathcal{E}}} u_2 \text{ then there exists a term } v \text{ such that } u_i \to^*_{\underline{\mathcal{E}}} v \ (i=1,2) \tag{2}$$

Indication of proof. On the basis of Lemma 3, we use the proof of Theorem 4 in [40] (with $\sim_{\underline{\mathcal{E}}}$ for $\equiv$) for the proof of this lemma. In essence, we prove (3) from (2) by induction on $\delta_{\underline{\mathcal{E}}}(u,v)$:

$$\text{If } u \sim_{\underline{\mathcal{E}}} v,\ u \to^*_{\underline{\mathcal{E}}} u' \in NF(\mathcal{E},\succ), \text{ and } v \to^*_{\underline{\mathcal{E}}} v' \in NF(\mathcal{E},\succ), \text{ then } u' = v'. \tag{3}$$

For the case where $\delta_{\underline{\mathcal{E}}}(u,v) = 0$ (i.e., $u = v$), the argument goes as follows. Suppose (3) false. We may suppose by (iii) that u is a

minimal term in {u: u falsifies (3) with $\underline{u}$ for $\underline{v}$} with respect to $\succeq$. Now $u \notin NF(\mathcal{E},\succ)$, because otherwise $u' = u = v'$. Therefore $u \neq u'$, $u \neq v'$, and there exist terms u_1, v_1 such that $u \rightarrow_{\underline{\mathcal{E}}} u_1 \rightarrow^*_{\underline{\mathcal{E}}} u'$ and $u \rightarrow_{\underline{\mathcal{E}}} v_1 \rightarrow^*_{\underline{\mathcal{E}}} v'$. By the lattice condition there is a term t such that $u_1 \rightarrow^*_{\underline{\mathcal{E}}} t$ and $v_1 \rightarrow^*_{\underline{\mathcal{E}}} t$, and by (iii) there is a term t' such that $t \rightarrow^*_{\underline{\mathcal{E}}} t' \in NF(\mathcal{E},\succ)$. However, (ii) and (iv) imply that $u \succ u_1$ and $u \succ v_1$, whence minimality of u implies that $u' = t' = v'$, a contradiction.

The induction step is straightforward.∎

Thus Proposition 2 reduces to the following:

Lemma 5. Suppose that (1) holds for each equation [u=v] in $\mathcal{E}$. Then $\underline{\mathcal{E}}$ satisfies the "lattice condition" (2).

Indication of proof. Suppose $u \rightarrow_{\underline{\mathcal{E}}} u_1$ and $u \rightarrow_{\underline{\mathcal{E}}} u_2$.

Case 1: $u = u'[r_1,r_2]$ where $u_1 = u'[t_1\theta_1,r_2]$, $u_2 = u'[r_1,t_2\theta_2]$ and $\mathcal{E}^{\sim}$ contains $[s_i=t_i]$ where $r_i = s_i\theta_i$ $(i=1,2)$. The conclusion follows with $v = u'[t_1\theta_1,t_2\theta_2]$.

Case 2: $u = u'[r_1'[r_2']]$ where $u_1 = u'[t_1\theta_1]$, $u_2 = u'[r_1'[t_2\theta_2]]$, and $\mathcal{E}^{\sim}$ contains $[s_1=t_1]$ where $r_1'[r_2'] = s_1\theta_1$, $[s_2=t_2]$ where $r_2' = s_2\theta_2$.

The details of this case are spelled out in [40,§5], and need not be repeated here. In each of two subcases (one of which uses (1)), it is shown that the consequent of (2) holds.∎

C.5 A D-Complexity Ordering

Let V be any vocabulary where $+,\cdot \in V_F^2$, + precedes $\cdot$, and let $\mathcal{E}$ be any axiom system which includes $D = \{D1,D2,D3,D4\}$:

D1. $[(x+y)+z = x+(y+z)]$

D2. $[(x\cdot y)\cdot z = x\cdot(y\cdot z)]$

D3. $[(x+y)\cdot z = z\cdot x + z\cdot y]$[1]

D4. $[z\cdot(x+y) = z\cdot x + z\cdot y]$

Define $\rightarrow_D$, $\rightarrow_D^*$, and NF(D) as in §2.3.2. (Briefly, $u[s] \rightarrow_D u[t]$ for all $[s=t]$ in D^*; $\rightarrow_D^*$ is the transitive, reflexive extension of $\rightarrow_D$ to $\mathcal{T}_V \cup \mathcal{C}_V$; and NF(D) is the set of $\rightarrow_D^*$-irreducible terms and clauses.)

We show below that NF(D) is a D-normal form for $\mathcal{T}_V$ (§1.2.9) The analysis of $\rightarrow_D^*$ which follows is assisted by two functions # and ¢: $\mathcal{T}_V \rightarrow N$:

$$\#(u) =_{df} [\text{If } u = (u_1+u_2) \text{ then } \#(u_1) + \#(u_2) + 1 \text{ else if } u = (u\cdot u_2) \text{ then } [\#(u_1)+1]\cdot[\#(u_2)+1] - 1 \text{ else } 0]$$

---

[1] In the usual axiomatizations for commutative rings, D3 is replaced by D_3': $[x+y)\cdot z = x\cdot z + y\cdot z]$, which is equivalent to D3 in the presence of $[(x\cdot y) = (y\cdot x)]$. The "Church-Rosser Theorem" for $\rightarrow_D^*$ below does not hold when D3 is replaced by D_3'.

$$\begin{aligned}\phi(u) =_{df} [&\text{If } u = (v_1+v_2)\\ &\text{then } \phi(v_1) + \phi(v_2)\\ &\text{else if } u = v_1 \cdot v_2\\ &\quad\text{then } \phi(v_1) + \phi(v_2) + \#(u)\\ &\quad\text{else if } u \in V_F^o \cup V_I\\ &\qquad\text{then } 0\\ &\qquad\text{else } \Sigma(\phi(u_i): i \le i \le n)\\ &\qquad\text{where } u = fu_1 \cdots u_n]\end{aligned}$$

Remark. It can be shown by structural induction on u that $u = u_o \to_D \cdots \to_D u_m = (v_o + \cdots + (v_{n-1} + v_n)\cdots) \in NF(D)$ where $n = \#(u)$ and v_k is not of the form $(s+t)$ $(k=0,\ldots,n)$, and that $\phi(u)$ is the number of {D3,D4}-contractions $(u_k \to_D u_{k+1})$ in this sequence. (If $\#(u) = 0$ then $u_m = v_o$ above.)

Let w be a weighting function for V wherein $w(\cdot) = w(+)$, and let $\succ_w$ be as defined in §C.1. The following relations are easily verified:

$$\begin{aligned}&(r\cdot s)\cdot t \succ_w r\cdot(s\cdot t)\ ;\\ &(r+s)+t \succ_w r+(s+t)\ ;\\ &t\cdot r+t\cdot s \succ_w (r+s)\cdot t\ ;\\ &t\cdot r\ t\cdot s \succ_w t\cdot(r+s)\ .\end{aligned} \tag{4}$$

Lemma 6. Suppose $u \to_D v$. Then $\#(u) = \#(v)$, and either $\phi(u) = \phi(v) + 1$ or else $\phi(u) = \phi(v)$ and $u >_w v$.

Indication of proof. The relation $(\#(u) = \#(v))$ is easily verified for each equation $[u=v]$ in D^* by expanding the definitions of

#(u) and #(v). It follows by structural induction on u that #(u) = #(v) for all u,v such that $u \rightarrow_D v$.

Similarly, we have $\phi(t\cdot(r+s)) = \phi((r+s)\cdot t) = \phi(t\cdot r + t\cdot s) + 1$, $\phi((r\cdot s)\cdot t) = \phi(r\cdot(s\cdot t))$, and $\phi((r+s)+t) = \phi(r+(s+t))$. From these relations and (4), the conclusion follows by structural induction on u. ∎

Corollary 1. For each term u, $u \rightarrow_D^* v$ for some term v in NF(D). Thus, NF(D) is a D-normal form for $\mathscr{T}_V$.

Corollary 2. $\rightarrow_D^*$ is an invariant partial ordering which satisfies D.C.C.. Thus, D is finitary.

Lemma 7: D satisfies (1) (with D for $\mathscr{E}$, $(D,\rightarrow_D^*)$ for $\underline{\mathscr{E}}$), whence D is canonical.

Proof. Verification that D satisfies (1) is straightforward, and is omitted. It follows by Proposition 2 that D is canonical. ∎

Define ν on $\mathscr{T}_V$ by

$\nu(u) =_{df}$ (the unique term v in NF(D) such that $u \rightarrow_{\underline{\mathscr{E}}}^* v$).

Let Q: $\mathscr{T}_V \times \mathscr{T}_V \rightarrow \{0,1\}$ be any computable function satisfying (C6)-(C8). ((C6) and (C7) are satisfied by the smallest relation Q satisfying (C8)).

Now let $\underline{D} = (D, \succ_W, \phi, Q)$.

Proposition 8. $\underline{D}$ satisfies (C1)-(C8) with $\underline{D}$ for $\underline{\mathscr{E}}$. The proof (based on Proposition 2) is fairly straightforward, and is omitted. By Proposition 1 we have the following:

Corollary. $\succ_{\underline{D}}$ is a D-complexity ordering.

D. A NORMAL REFINEMENT FOR INTEGER ARITHMETIC

Integer Arithmetic is the first-order theory of ordered integral domains satisfying each induction axiom

$$I_C\colon\ C[0/x] \wedge \forall x\ (NNx \wedge C \rightarrow C[(x+1)/x]) \rightarrow \forall x(NNx \rightarrow C)$$

where C is a formula having extra-logical constants in $\{NN, 0, 1, -, +, \cdot\}$. NN represents the set of non-negative integers or natural numbers.

In practical theorem-proving applications we include only those induction axioms deemed necessary for the problem at hand. Thus, we work with finite extensions of an axiomatization IA for the theory of ordered integral domains having no elements between zero (0) and unity (1).

The existence of quotient and remainder operations (as shown by the classical Euclidean algorithm) can be derived from $IA \cup \{I_C\colon\ C \in \mathcal{F}_V\}$. The clause-representation of the "Euclidean algorithm theorem"

$$\forall_w\ \forall_x\ \forall_y\ \forall_z(NNz \wedge NN(|y| - z) \wedge w \cdot y + z = x)$$

introduces a Skolem operator ($\div$) representing division. By the addition of this and other relevant theorems as lemmas, the basic axiomatization IA is extended so that increasingly advanced results in number theory can be stated and perhaps proved without further induction axioms. Ideally, each such extension is supplied with a corresponding macro-refinement which can be composed with a previously defined normal refinement for the theory being extended.

In this appendix, a normal refinement for a simple extension of IA is defined and used by a normal proof procedure to generate a refutation proof for the following classical result in number theory:

The square root of a prime number is irrational. (1)

The refutation itself contains 49 clauses. Even with an "optimal" search strategy, the normal proof procedure generates 230 clauses. Of these, 52 can be eliminated by simplification and subsumption operations as soon as they are generated. Thus, the "efficiency ratio" of the proof procedure on this problem is better than 0.2 (49/230 or 49/178; see §B).

The normal refinement has the form

$$\Delta = HR(\mathcal{E}_1, \succ_D, s) \cdot ND(\mathcal{E}_0, \succ_D)$$

where HR and ND are as defined in §2, $\succ_D$ is a D-complexity ordering (§C.1), and $D = \{D1, D2, D3, D4\} \subseteq IA$:

D1: $[(x+y) + z = x + (y+z)]$;

D2: $[(x \cdot y) \cdot z = x \cdot (y \cdot z)]$;

D3: $[(x+y) \cdot z = z \cdot x + z \cdot y]$; and

D4: $[z \cdot (x+y) = z \cdot x + z \cdot y]$.

The D-complexity ordering $\succ_D$ is of independent interest, in that it makes each member of D a reduction (e.g., $z \cdot (x+y) \succ_D z \cdot x + z \cdot y$); moreover, the refinement $ND(\mathcal{E}, \succ_D)$ can be used with any axiomatization $\mathcal{E}$ where $\mathcal{E} \supseteq D$. Define $\mathcal{I}_D$ by

$$\mathcal{J}_D =_{df} \{u \in \mathcal{J}_V: u \text{ contains no subterm of the form } (r+s) + t,\ (r\cdot s)\cdot t, (r+s)\cdot t, \text{ or } t\cdot(r+s)\}\ .$$

The restriction of $\succ_D$ to $\mathcal{J}_D$ is essentially a "syntactical" complexity ordering $\succ_W$ as defined in §C.2.

Remark. It may well be that $\succ_W$ and numerous other complexity orderings yield refinements which perform just as well or better than Δ on the problem-domain $\models_{IA}$. My preference for $\succ_D$ is based on the existence of a useful IA-canonical form $\mathcal{J}_C \subseteq \mathcal{J}_D$ for terms over the vocabulary V where $V_F = \{+,\cdot,0,1\}$. Each member of $\mathcal{J}_C$ has the form $(t_1 + \cdots + (t_{n-1} + t_n)\cdots)$ where t_i has the form $(x_1 \cdot \cdots \cdot (x_{n-1} \cdot x_n)\cdots)$ or $-(x_1 \cdot \cdots \cdot (x_{n-1} \cdot x_n)\cdots)$ and $t_i \neq (-t_j)$ $(i,j \in \{1,\cdots,n\})$.

A normal proof procedure based on Δ computes increasingly stronger IA-normal forms $NF(\mathcal{E},\succ_D)$ where $\mathcal{E} \supseteq IA$ and hence $\mathcal{J}_C \subseteq NF(\mathcal{E},\succ_D) \subseteq NF(D)$. Thus, such a proof procedure "tends" to keep terms in a convenient IA-canonical form. (See §C.0 (Efficient representations.))

D.1 Integer Arithmetic

The clause-form axiomatization IA(I1-I15 below) formalizes the concept of an ordered integral domain[1] $\underline{I} = (I,J,+,\cdot,-,0,1)$ having no elements between zero (0) and unity (1):

[1] The present treatment of Integer Arithmetic is based on "The Integers and Integral Domains," Chapter 4 in Solomon Feferman's The Number Systems (Addison-Wesley Publishing Co., Inc., Palo Alto, California, 1964). However, IA is not the axiom system used in Feferman's development.

I1: $[0+x = x]$;

I2: $[x+(-x) = 0]$;

I3: $[(x+y)+z = x+(y+z)]$;

I4: $[0 \cdot x = 0]$;

I5: $[1 \cdot x = x]$;

I6: $[(x \cdot y) \cdot z = x \cdot (y \cdot z)]$;

I7: $[(x+y) \cdot z = z \cdot x + z \cdot y]$;

I8: $[z \cdot (x+y) = z \cdot x + z \cdot y]$:

I9: $[x+y = y+x]$;

I10: $[x \cdot y = y \cdot x]$;

I11: $[0 \neq 1]$;

I12: $\underline{[x \cdot y \neq 0]} \vee [x=0] \vee [y=0]$;

I13: $\underline{\sim NNx} \vee \sim NNy \vee NN(x+y)$;

I14: $\underline{\sim NNx} \vee \sim NNy \vee NN(x \cdot y)$;

I15: $\sim NN{-x} \vee \sim NNx \vee [x=0]$;

I16: $\underline{NN(x+(-1))} \vee NN(-x)$.

Commutative Ring with Unity

Integral Domain

Ordered Integral Domain with Nothing between 0 and 1

Clearly IA is valid when interpreted into the "standard" system of integers defined in our metalanguage. On the other hand, IA is valid in many "nonstandard" structures wherein the integers can be isomorphically embedded[2].

---

[2]It is well known that no decidable set of axioms can characterize the integers up to isomorphism. For an example of a system $\underline{I}$ which satisfies IA but not $\{I_C: C \in \mathcal{F}_V\}$, let I be the set of all "canonical form" polynomials in a single variable z with integer coefficients, stipulating that $n < z \leq z^n$ for each $n > 0$.

D.2 A Formalization of the Irrational Prime Root Theorem

Our first task is to express (1) in the language of Integer Arithmetic. Define the predicate PN (Prime Number) by

$$PN(z) \leftrightarrow [z \neq 0] \wedge [z \neq 1] \wedge [z \neq -1]$$
$$\forall x \forall y(\exists w[z \cdot w = x \cdot y] \rightarrow \exists w([z \cdot w = x] \vee [z \cdot w = y])) \quad (2)$$

The Irrational Prime Root Theorem (1) can now be expressed by

$$\forall z(PNz \rightarrow \forall x \forall y(z \cdot y \cdot y \neq x \cdot x)) \quad (3)$$

To see this, observe that every rational number has the form $[x/y]$ where x and y are integers, and that $z = [x/y] \cdot [x/y]$ iff $z \cdot y \cdot y = x \cdot x$.

Unfortunately, (3) appears <u>not</u> to be a logical consequence of (IA) $\cup$ (2), for the reason that instances of the induction schema I_C are needed to prove that if $a\ c \cdot c = b \cdot b$ then there exist <u>relatively prime</u> b', c' such that $a \cdot c' \cdot c' = b' \cdot b'$. This existence lemma is a sufficiently difficult sub-problem to make the automatic generation of a proof for (3) from (IA) $\cup \{I_C\} \cup$ (2) a virtually impractical task, even with the best currently available proof procedures.

<u>Remark</u>. This direct but arduous approach would essentially involve proving the "Euclidean algorithm theorem",

$$\forall w \forall x[NN(x + (-1) \rightarrow \exists y \exists z([x \cdot y + z = w] \wedge NNz \wedge NNx + (-z))] \quad (4)$$

Now the Skolem functions for y and z in the clause form of (4) are familiar arithmetical operators, $\div$ and mod :

$$\sim NN(x+(-1)) \vee [x\cdot(w \div x) + (w \bmod x) = w] \ ;$$

$$\sim NN(x+(-1)) \vee NN(w \bmod x) \ ;$$

$$\sim NN(x+(-1)) \vee NN(x+-(w \bmod x) + (-1)) \ . \qquad (5)$$

$(w \div x)$ is the integer quotient of w and x, and $(w \bmod x)$ is the residue of w modulo x --i.e., the "remainder" of $(w \div x)$.

Having sufficiently appreciated the conniving which frequently passes under the guise of "formalization" in a basically impractical "automatic theorem-proving" task, we extend IA with the concepts of quotient and, for brevity's sake, divides:

$$\sim Dxy \vee [x\cdot[y \div x] = y] \qquad (6)$$

Returning to the previously noted difficulty with (3), we avoid it simply by "minimizing" x and y to relatively prime numbers in (3):

$$\forall x(PNz \rightarrow \forall x \forall y(\forall w(Dwx \wedge Dwy \rightarrow [w=1]) \rightarrow z\cdot y\cdot y \neq x\cdot x))) \qquad (7)$$

The new theorem to be proved is (7). It is not difficult to guess that (7) follows from IA augmented by (6) and the following clauses obtained from (2):

$$\sim(PNz) \vee \sim(Dz(x\cdot y)) \vee (Dzx) \vee (Dzy) \ ;$$

$$\sim PN0;$$

$$\sim PN1. \qquad (8)$$

Remark. $\exists w(z\cdot w = x\cdot y)$ in (2) is translated into $Dz(x\cdot y)$ in (8), and (6) will eventually translate a positive literal Duv into

$[u \cdot [v \div u] = v]$ during the proof of (7). We could have used the direct "clause-representation" translation on $\exists w(z \cdot w = x \cdot y)$, introducing a new Skolem function f and translating $\exists w(z \cdot w = x \cdot y)$ into $[z \cdot f(z,x,y) = x \cdot y]$. This would have complicated the proof of (7) in §D.5 only slightly.

To conclude this ad hoc formalization process, we note that (IA) is to be augmented by (6), (8), and the following clause-representation for the negation of (7):

$$\begin{aligned} &PNa\ ; \\ &\sim Dwb \vee \sim Dwc \vee [w=1]\ ; \\ &[a \cdot c \cdot c = b \cdot b]\ . \end{aligned} \tag{9}$$

Let $\mathcal{E} =_{df} (IA \cup (6) \cup (8))$. $\mathcal{E}$ is allegedly consistent, and $\mathcal{E} \cup (9)$ is allegedly inconsistent. Our next task is to define a good normal refinement for $\mathcal{E}$.

D3. A Normal Refinement for Extended Integer Arithmetic

Thus far we have defined an extension $\mathcal{E} = (IA \cup (6) \cup (8))$ for IA. Next we define a complexity order $\succ_D$ for use in a resolution micro-refinement $ND(\mathcal{E}, \succ_D)$. The vocabulary V of $\mathcal{E} \cup (9)$ contains nine operation constants, to which we assign the following weights in accordance with §2.3.5:

Constant	Degree	Weight
0	0	1
1	0	1
a	0	1
b	0	1
c	0	1
+	2	0
•	2	0
÷	2	0
-	1	0

Fig. 2. A weighting function

Let w be the weighting function defined by Figure 2, and let $<$ be the well-ordering of constants wherein $0 < 1 < a < b < c < + < \cdot < \div < -$. Define the complexity ordering $\succ_W$ on $\mathcal{T}_V$ as in §2.3.5.

Remark. Observe that - is assumed to be the last element of V_F , and hence that $w(-)$ can be equated to 0 .

Now consider the clauses of (IA). Equations I1-I6 are reductions according to $\succ_W$, in the sense that the left-hand side is more complex than the right-hand side. However, for I7 we have $(z \cdot x) + (z \cdot y) \succ_W (x+y) \cdot z$ because $w((z \cdot x) + (z \cdot y)) = 4$ and $w((x+y) \cdot z) = 3$. Having a strong intuitive feeling that any good complexity ordering for $\mathcal{E}$ should treat I1-I8 as reductions, I looked for a "natural modification" of $\succ_W$ wherein I1-I8 would all be reductions. The resulting relation $\succ_D$ (§C.5) is defined first on a D-canonical form $\mathcal{T}_D$ for $\mathcal{T}_V$ and then extended to $\mathcal{T}_V$; $\succ_D$ agrees with $\succ_W$ on $\mathcal{T}_D$. It follows by Proposition 8 in §C.5 that $\succ_D$ is a D-complexity ordering.

Now let $\mathcal{E}_1$ = (I1-I11) $\cup$ (I13-I15) $\cup$ (6) $\cup$ (9), and let $\mathcal{E}_0$ = (I1-I11) $\cup$ {$\sim$PNo, $\sim$PN1, PNa, [a•(c•c) = b•b]} . Let Δ = HR($\mathcal{E}_1$,$\succ_D$,s) •ND($\mathcal{E}_0$,$\underline{\underline{\succ_D}}$) where r_s is the identity renaming and, if C is non-positive and non-null, then s(C) is a negative literal whose atom is maximal among atoms of negative literals in C (with respect to $\succ_D$).

The refutation which follows (Figure 3) is a complete refutation Δ(IA $\cup$ (6) $\cup$ (8) $\cup$ (9))$^-$.

D.4 A Complete Refutation

The integer domain axiom I11 is clearly necessary for the proofs of many basic theorems of Integer Arithmetic. Unfortunately, the two positive equations in this clause render the presently available completeness results on normal refinements inapplicable to these problems which depend upon it. As usual, formally verified knowledge lags behind intuitively justified beliefs. Perhaps, after seeing numerous machine-generated refutations based on normal refinements not guaranteed to be complete by present results, we will begin to see how to extend these results.

The formats for the refutations which follow are the same as the format of Figure 1 in §C.2. Figure 2 is the proof tree used to guide the order of selection of clauses from the queue (Q) in Figure 3. While a normal proof procedure would not typically have access to this "ideal" search strategy, Figure 3 is nevertheless informative insofar as it shows how sparse the search space can be when Δ (§D.3) is used to find a refutation tree containing 49 clauses.

The Square Root of a Prime Number is Irrational

(Proof Tree Only)

In	Clause	Source
*0.	$[1 \neq 0]$	Ax (Not used)
1.	$0+x = x$	Ax
-2.	$-x+x = 0$	Ax
-3.	$(x+y)+z =: x+(y+z)$	Ax
4.	$0 \cdot x = 0$	Ax
*5.	$1 \cdot x = x$	Ax (Not used)
+6.	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	Ax
+7.	$w \cdot (x+y) = w \cdot x + w \cdot y$	Ax
+8.	$x+y = y+x$	Ax
+9.	$x \cdot y = y \cdot x$	Ax
+10.	$[x \cdot y \neq 0] \vee [x=0] \vee [y=0]$	Ax
+11.	$Dx(x \cdot y)$	Pr
+12.	$\sim Dxy \vee [x \cdot (y \div x) = y]$	Pr
+13.	$\sim PNx \vee \sim Dx(y \cdot z) \vee Dxy \vee Dxz$	Pr
+14.	$\sim PNo$	Pr
+15.	$\sim PN1$	Pr
+16.	PNa	NC
+17.	$[a \cdot (c \cdot c) = b \cdot b]$	NC
+18.	$\sim Dzb \vee \sim Dzc \vee [z=1]$	NC
+19.	$Da(b \cdot b)$	Rp(17,11)
+20.	$\sim Da(y \cdot z) \vee Day \vee Daz$	Cut(16,13)
+21.	Dab	Cut(19,20)
+22.	$a \cdot (b \div a) =: b$	Cut(21,12)
+23.	$b \cdot z = a \cdot ((b \div a) \cdot z)$	Rp(22,6)
+24.	$a \cdot ((b \div a) \cdot z) = b \cdot z$	Rp(CNR(23),[x=x])
+25.	$[z \cdot x + z \cdot y \neq 0] \vee [z=0] \vee [x+y=0]$	Rp(7,10)
+26.	$[a \cdot x + b \cdot z \neq 0] \vee [a=0] \vee [x + (b \div a) \cdot z = 0]$	Rp(24,25)
+27.	$[b \cdot b + b \cdot z \neq 0] \vee [c \cdot c + (b \div a) \cdot z = 0] \times [a=0]$	Rp(17,26)
+28.	$x + (-x) = 0$	Rp(8,2)

In	Clause	Source
+29.	$w \cdot 0 = w \cdot x + w \cdot (-x)$	Rp(28,7)
+30.	$w \cdot x + w \cdot (-x) = 0$	Rp(CNR(29),[x=x])
+31.	$[0 \neq 0] \vee [a=0] \vee [c \cdot c + (b \div a) \cdot (-b) = 0]$	Rp(30,27)
+32.	$[c \cdot c + (b \div a) \cdot (-b) = 0] \vee [a=0]$	Cut([x=x],31)
+33.	$[0+z = c \cdot c + ((b \div a) \cdot (-b) + z] \vee [a=0]$	Rp(32,3)
+34.	$[c \cdot c + ((b \div a) \cdot (-b) + z) = z] \vee [a=0]$	Rp(CNR(33),[x=x])
+35.	$w \cdot (-x) + w \cdot x = 0$	Rp(8,30)
+36.	$[(c \cdot c + 0) = (b \div a) \cdot b] \vee [a=0]$	Rp(35,34)
+37.	$[(b \div a) \cdot b = c \cdot c] \vee [a=0]$	Rp(CNR(36),[x=x])
+38.	$Dw(x \cdot w)$	Rp(9,11)
+39.	$Dw(x \cdot (y \cdot w))$	Rp(6,38)
+40.	$Dw(x \cdot (w \cdot y))$	Rp(9,39)
+41.	$Da(x \cdot b)$	Rp(22,40)
+42.	$Da(c \cdot c) \vee [a=0]$	Rp(37,41)
+43.	$Dac \vee [a=0]$	Cut(42,20)
+44.	$\sim Dac \vee [a=1]$	Cut(21,18)
+45.	$[a=0] \vee [a=1]$	Cut(43,44)
+46.	$PN1 \vee [a=0]$	Rp(45,16)
+47	$[a=0]$	Cut(46,15)
+48.	PNo	Rp(47,16)
49.	0	Cut(48,14)

(Proof using search strategy described in §A.2)

In	Clause	Source	Out		Disposition
+1.	$0+x = x$	Ax	1		
+2.	$-x+x = 0$	Ax	2		
+3.	$(x+y)+z = x+(y+z)$	Ax	3		
+4.	$0 \cdot x = 0$	Ax	4		
+5.	$1 \cdot x = x$	Ax	5		
+6.	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	Ax	6		
+7.	$w \cdot (x+y) = w \cdot x + w \cdot y$	Ax	7		
+8.	$x+y = y+x$	Ax	8		
+9.	$x \cdot y = y \cdot x$	Ax	9		
+10.	$[x \cdot y \neq 0] \vee [x=0] \vee [y=0]$	Ax	20		
+11.	$Dx(x \cdot y)$	Pr	11		
+12.	$\sim Dxy \vee [x \cdot (y \div x) = y]$	Pr	16		
+13.	$\sim PNx \vee \sim Dx(y \cdot z) \vee Dxy \vee Dxz$	Pr	13		
+14	$\sim PNo$	Pr	46		
+15.	$\sim PN1$	Pr	43		
+16.	PNa	NC	41		
+17	$a \cdot (c \cdot c) = b \cdot b$	NC	10		
+18.	$\sim Dzb \vee \sim Dzc \vee [z=1]$	NC	39		
19.	$y+z = 0+(y+z)$	Rp(1,3)		19	CNR(*)
20.	$0+z = x+(x+z)$	Rp(2,3)		20	CNR(+)
21.	$(w+(x+y))+z = (w+x)+(y+z)$	Rp(3,3)		21	CNR(*)
22.	$0 \cdot z = 0 \cdot (y \cdot z)$	Rp(4,6)		22	CNR(*)
23.	$y \cdot z = 1 \cdot (y \cdot z)$	Rp(5,6)		23	CNR(*)
24.	$(w \cdot (x \cdot y)) \cdot z = (w \cdot x) \cdot (y \cdot z)$	Rp(6,6)		24	CNR(*)
25.	$w \cdot y = w \cdot 0 + w \cdot y$	Rp(1,7)		25	CNR(*)
26.	$w \cdot 0 = w \cdot (-x) + w \cdot x$	Rp(2,7)		26	CNR(*)
27.	$w \cdot (x+(y+z)) = w \cdot x + w \cdot (y+z)$	Rp(3,7)		27	Sub(7)
28.	$0 = 0 \cdot x + 0 \cdot y$	Rp(4,7)		28	CNR(*)
29.	$(x+y) = 1 \cdot x + 1 \cdot y$	Rp(5,7)		29	CNR(*)

In	Clause	Source	Out		Disposition
30.	$w\cdot(z\cdot(x+y)) = (w\cdot z)\cdot x + (w\cdot z)\cdot y$	Rp(6,7)		30	CNR(+)
31.	$x+0 = x$	Rp(8,1		31	
+32.	$x + (-x) = 0$	Rp(8,2)	22		
33.	$(y+x)+z = x + (y+z)$	Rp(8,3)		33	CNR(+)
34.	$z + (x+y) = x + (y+z)$	Rp(8,3)			
35.	$w + (x+y) = y + (w+x)$	Rp(3,8)			
36.	$w\cdot(y+x) = w\cdot x + w\cdot y$	Rp(8,7)		36	CNR(*)
37.	$x\cdot 0 =: 0$	Rp(9,4)			
38.	$x\cdot 1 =: x$	Rp(9,5)			
39.	$(y\cdot x)\cdot z = x\cdot(y\cdot z)$	Rp(9,6)			CNR(+)
40.	$z\cdot(x\cdot y) = x\cdot(y\cdot z)$	Rp(9,6)			
41.	$w\cdot(x\cdot y) = y\cdot(w\cdot x)$	Rp(6,9)			
42.	$(x+y)\cdot w = w\cdot x + w\cdot y$	Rp(9,7)			CNR(+)
43.	$(b\cdot b)\cdot z = a\cdot((c\cdot c)\cdot z)$	Rp(17,6)			CNR(+)
44.	$(c\cdot c)\cdot a = b\cdot b$	Rp(9,17)			CNR(*)
45.	D00	Rp(4,11)			
46.	D1y	Rp(5,11)			
47.	$D(w\cdot x)(w\cdot(x\cdot y))$	Rp(6,11)			
48.	$Dw(w\cdot x + w\cdot y)$	Rp(7,11)			
+49.	$Dx(y\cdot x)$	Rp(9,11)		33	
+50.	$Da(b\cdot b)$	Rp(17,11)		12	
+51.	$\sim Da(y\cdot z)$ v Day v Daz	Cut(16,13)		14	
52.	$\sim Da0$ v Da0 v Daz	Rp(4,51)			
53.	$\sim Daz$ v Da1 v Daz	Rp(5,51)			
54.	$\sim Da(x\cdot(y\cdot z))$ v $Da(x\cdot y)$ v Daz	Rp(6,51)			
55.	$\sim Da(w\cdot x + w\cdot y)$ v Daw v $Da(x+y)$	Rp(9,51)			
56.	$\sim Da(z\cdot y)$ v Day v Daz	Rp(9,51)			
57.	$\sim Da(b\cdot b)$ v Daa v $Da(c\cdot c)$	Rp(17,51)			
+58.	Dab	Cut(50,51)		15	
+59.	$a\cdot(b \div a) =: b$	Cut(58,12)		17	
+60.	$b\cdot z = a\cdot((b \div a)\cdot z)$	Rp(59,6)		18	CNR(*)
61.	$(b \div a)\cdot a = b$	Rp(9,59)			
62.	Dab	Rp(59,11)			Sub(58)
63.	$\sim Dab$ v Daa v $Da(b \div a)$	Rp(59,51)			

In	Clause	Source	Out	Disposition
+64.	$a \cdot ((b \div a) \cdot z) = b \cdot z$	Rp(60,[x=x])	19	
65.	$((b \div a) \cdot z) \cdot a = b \cdot z$	Rp(9,64)		CNR(+)
66.	$a \cdot (z \cdot (b \div a)) = b \cdot z$	Rp(9,64)		
67.	$[0 \neq 0] \vee [0=0] \vee [y=0]$	Rp(4,10)		Sub(x=x)
68.	$[y \neq 0] \vee [1=0] \vee [y=0]$	Rp(5,10)		Tautology
69.	$(w \cdot (x\ y) \neq 0 \vee [w \cdot x = 0] \vee [y=0]$	Rp(6,10)		
+70.	$[w \cdot x + w \cdot y \neq 0] \vee [w=0] \vee [x+y=0]$	Rp(7,10)	21	
71.	$[y \cdot x \neq 0] \vee [x=0] \vee [y=0]$	Rp(9,10)		
72.	$[b \cdot b \neq 0] \vee [a=0] \vee [c \cdot c = 0]$	Rp(17,10)		
73.	$[b \neq 0] \vee [a=0] \vee [(b \div a) = 0]$	Rp(59,10)		
74.	$[b \cdot z \neq 0] \vee [a=0] \vee [(b \div a) \cdot z = 0]$	Rp(64,10)		
75.	$[0+0 \cdot y \neq 0] \vee [0=0] \vee [x+y=0]$	Rp(4,70)		Sub([x=x])
76.	$[0 \cdot x+0 \neq 0] \vee [0=0] \vee [x+y=0]$	Rp(5,70)		Sub([x=x])
77.	$[w \cdot (z \cdot x) + (w \cdot z) \cdot y \neq 0] \vee [w \cdot z = 0] \vee [x+y = 0]$	Rp(6,70)		CNR(+)
78.	$[(w \cdot z) \cdot x + (w \cdot (z \cdot y)) \neq 0] \vee [w \cdot z = 0] \vee [x+y = 0]$	Rp(6,70)		CNR(*)
79.	$[(w \cdot x_1 + w \cdot x_2) + w \cdot y \neq 0] \vee [w=0] \vee [(x_1+x_2)+y = 0]$	Rp(7,70)		CNR(+)
80.	$[w \cdot x + (w \cdot y_1 + w_1 y_2) \neq 0] \vee [w=0] \vee [x + (y_1+y_2) = 0]$	Rp(7,70)		CNR(+)
81.	$[w \cdot y + w \cdot x \neq 0] \vee [w=0] \vee [x+y=0]$	Rp(8,70)		
82.	$[x \cdot w + w \cdot y \neq 0] \vee [w=0] \vee [x+y=0]$	Rp(9,70)		
83.	$[w \cdot x + y \cdot w \neq 0] \vee [w=0] \vee [x+y=0]$	Rp(9,70)		
84.	$[b \cdot b + a \cdot y \neq 0] \vee [a=0] \vee [c \cdot c+y=0]$	Rp(17,70)		
85.	$[a \cdot x + b \cdot b \neq 0] \vee [a=0] \vee [x+c \cdot c=0]$	Rp(17,70)		
86.	$[b+a-y \neq 0] \vee [a=0] \vee [(b \div a)+y = 0]$	Rp(59,70)		
87.	$[a \cdot x + b \neq 0] \vee [a=0] \vee [x+(b \div a)=0]$	Rp(59,70)		
88.	$[b \cdot z + a\ y \neq 0] \vee [a=0] \vee [(b \div a) \cdot z + y = 0]$	Rp(64,70)		
+89.	$[a \cdot x + b \cdot z \neq 0] \vee [a=0] \vee [x + (b \div a) \cdot z = 0]$	Rp(64,70)		
90.	$[b \cdot z + a \cdot x \neq 0] \vee [a=0] \vee [x + (b \div a) \cdot z = 0]$	Rp(8,89)		
91.	$[x \cdot a + b \cdot z \neq 0] \vee [a=0] \vee [x + (b \div a) \cdot z = 0]$	Rp(9,89)		

In	Clause	Source	Out	Disposition
92.	$[a\cdot x + z\cdot b \neq 0] \vee [a=0] \vee [x+(b \div a)\cdot z = 0]$	Rp(9,89)		
+93.	$[b\cdot b + b\cdot z \neq 0] \vee [a=0]$ $\vee [c\cdot c + (b\div a)\cdot z=0]$	Rp(17,89)	25	
94.	$[b+b\cdot z \neq 0] \vee [a=0]$ $\vee [(b \div a) + (b \div a)\cdot z = 0]$	Rp(59,89)		
95.	$[b\cdot w + b\cdot z \neq 0] \vee [a=0]$ $\vee [(b \div a)\cdot w + (b \div a)\cdot z = 0]$	Rp(64,89)		
96.	$-0 = 0$	Rp(1,32)		
97.	$x + (y + (-(x+y))) = 0$	Rp(3,32)		
98.	$0 + z = x + ((-x)+z)$	Rp(32,3)		CNR(+)
+99.	$w\cdot 0 = w\cdot x + w\cdot(-x)$	Rp(32,7)	23	CNR(*)
100.	$(-x) + x = 0$	Rp(8,32)		
+101.	$w\cdot x + w\cdot(-x) = 0$	Rp(NR(99), [x=x])	24	
102.	$0+z = w\cdot x + (w\cdot(-x) + z)$	Rp(101,3)		CNR(+)
103.	$0+0\cdot(-x) = 0$	Rp(4,101)		CNR(*)
104.	$0\cdot x+0 = 0$	Rp(4,101)		CNR(*)
105.	$x + 1\cdot(-x) = 0$	Rp(5,101)		CNR(*)
106.	$1\cdot x + (-x) = 0$	Rp(5,101)		CNR(*)
107.	$w\cdot(x\cdot y) + (w\cdot x)\cdot(-y) = 0$	Rp(6,101)		CNR(+)
108.	$(w\cdot x)\cdot y + w\cdot(x\cdot(-y)) = 0$	Rp(6,101)		CNR(*)
109.	$(w\cdot x_1 + w\cdot x_2) + w\cdot(-(x_1+x_2)) = 0$	Rp(7,101)		CNR(+)
+110.	$w\cdot(-x) + w\cdot x = 0$	Rp(8,101)	30	
111.	$x\cdot w + w\cdot(-x) = 0$	Rp(9,101)		
112.	$w\cdot x + (-x)\cdot w = 0$	Rp(9,101)		
113.	$b\cdot b + a\cdot(-(c\cdot c)) =: 0$	Rp(17,101)		
114.	$b + a\cdot(-(b \div a)) = 0$	Rp(59,101)		
115.	$b\cdot z + a\cdot(-((b \div a)\cdot z))$	Rp(64,101)		
116.	$[b\cdot b + (b\cdot x + b\cdot y) \neq 0] \vee [a=0]$ $\vee [c\cdot c + (b \div a)\cdot(x+y) = 0]$	Rp(7,93)		
117.	$[b\cdot z + b\cdot b \neq 0] \vee [a=0]$ $\vee [c\cdot c + (b\div a)\cdot z=0]$	Rp(8,93)		
118.	$[b\cdot b + b\cdot z \neq 0] \vee [a=0]$ $\vee [c\cdot c + (b \div a)\cdot z = 0]$	Rp(9,93)		Sub(93)

In	Clause	Source	Out	Disposition
119.	$[b\cdot b + z\cdot b \neq 0] \vee [a=0] \vee [c\cdot c + (b \div a)\cdot z = 0]$	Rp(9,93)		
+120.	$[0 \neq 0] \vee [a=0] \vee [c\cdot c + (b \div a)\cdot(-b) = 0]$	Rp(101,93)	26	
+121.	$[c\cdot c + (b \div a)\cdot(-b) =: 0] \vee [a=0]$	Cut([x=x], 120)	27	
+122.	$[0+z = c\cdot c + ((b \div a)\cdot(-b) + z)] \vee [a=0]$	Rp(121,3)		
123.	$[(b \div a)\cdot(-b) + c\cdot c = 0] \vee [a=0]$	Rp(8,121)		
124.	$[c\cdot c + (b \div a)\cdot(-b) = 0] \vee [a=0]$	Rp(9,121)		Sub(121)
125.	$[c\cdot c + (-b)\cdot(b \div a) = 0] \vee [a=0]$	Rp(9,121)		
+126.	$[c\cdot c + ((b \div a)\cdot(-b) + z) = z] \vee [a=0]$	Rp(CNR(122), [x=x])	29	
127.	$[((b \div a)\cdot(-b) + z] \vee [a=0]$	Rp(8,126)		CNR(+)
128.	$[c\cdot c + (z + (b \div a)\cdot(-b)) = z] \vee [a=0]$	Rp(8,126)		
129.	$[c\cdot c + ((b \div a)\cdot(-b) + z) = z] \vee [a=0]$	Rp(9,126)		
130.	$[c\cdot c + ((-b)\cdot(b \div a) + z = z] \vee [a=0]$	Rp(9,126)		
131.	$[c\cdot c + 0 = (b \div a)\cdot(-(-b))] \vee [a=0]$	Rp(101,126)		CNR(+)
132.	$[0+z = w\cdot(-x) + (w\cdot x + z)]$	Rp(110,3)		CNR(+)
133.	$0 + 0\cdot x = 0$	Rp(4,110)		CNR(*)
134.	$0\cdot(-x) + 0 = 0$	Rp(4,110)		CNR(*)
135.	$-x + 1\cdot x = 0$	Rp(5,110		CNR(*)
136.	$1\cdot(-x) + x = 0$	Rp(5,110)		CNR(*)
137.	$w\cdot(y\cdot(-x)) + (w\cdot y)\cdot x = 0$	Rp(6,110)		CNR(+)
138.	$(w\cdot y)\cdot(-x) + w\cdot(y\cdot x) = 0$	Rp(6,110)		CNR(*)
139.	$w\cdot(-(y+z)) + (w\cdot y + w\cdot z) = 0$	Rp(7,110)		
140.	$w\cdot x + w\cdot(-x) = 0$	Rp(8,110)		
141.	$(-x)\cdot w + w\cdot x = 0$	Rp(9,110)		
142.	$w\cdot(-x) + x\cdot w = 0$	Rp(9,110)		
143.	$a\cdot(-(c\cdot c)) + b\cdot b = 0$	Rp(17,110)		
144.	$a\cdot(-(b \div a)) + b = 0$	Rp(59,110)		
145.	$a\cdot(-((b \div a)\cdot z)) + b\cdot z = 0$	Rp(64,110)		
146.	$[0 \neq 0] \vee [w=0] \vee [(-x) + x = 0]$	Rp(110,70)		
*147	$[c\cdot c + 0 = (b \div a)\cdot b] \vee [a=0]$	Rp(110,126)	31	
+148.	$[(b \div a)\cdot b = c\cdot c] \vee [a=0]$	Rp(CNR(147), [x=x])	32	

In	Clause	Source	Out	Disposition
149.	$[c\cdot(c\cdot z) = (b \div a)\cdot(b\cdot z)] \vee [a=0]$	Rp(148,6)		
150.	$[b\cdot(b \div a) = c\cdot c] \vee [a=0]$	Rp(9,148)		
151.	$[c\cdot c + (b \div a)\cdot y \neq 0] \vee [b \div a = 0] \vee [b+y = 0] \vee [a=0]$	Rp(148,70)		
152.	$[(b \div a)\cdot x + c\cdot c \neq 0] \vee [b \div a = 0] \vee [x+b = 0] \vee [a=0]$	Rp(148,70)		
153.	$[c\cdot c + (b \div a)\cdot(-b) =: 0] \vee [a=0]$	Rp(148,101)		
154.	Dx0	Rp(4,49)		
155.	Dxx	Rp(5,49)		
+156.	$Dx(y\cdot(w\cdot x))$	Rp(6,49)	34	
157.	$D(y+z)(x\cdot y + x\cdot z)$	Rp(7,49)		
158.	$Dx(x\cdot y)$	Rp(9,49)		Sub(11)
159.	$D(c\cdot c)(b\cdot b)$	Rp(17,49)		
160.	$D(b \div a)b$	Rp(59,49)		
161.	$D((b \div a)\cdot z)(b\cdot z)$	Rp(64,49)		
162.	$Db(c\cdot c) \vee [a=0]$	Rp(148,49)		
163.	Dx0	Rp(4,156)		Sub(154)
164.	$Dx(y\cdot 0)$	Rp(4,156)		CNR(*)
165.	$Dx(w\cdot x)$	Rp(5,156)		
166.	$Dx(y\cdot x)$	Rp(5,156)		Sub(165)
167.	$Dx(x_1\cdot(y\cdot(w\cdot x)))$	Rp(6,156)		
168.	$Dx(y\cdot(w_1\cdot(w_2\cdot x)))$	Rp(6,156)		
169.	$D(x+y)(y\cdot(w\cdot x + w\cdot y))$	Rp(7,156)		CNR(+)
170.	$Dx((w\cdot x)\cdot y)$	Rp(9,156)		CNR(x)
+171.	$Dx(y\cdot(x\cdot w))$	Rp(9,156)	35	
172.	$Dc(b\cdot b)$	Rp(17,156)		
173.	$D(c\cdot c)(y\cdot(b\cdot b))$	Rp(17,156)		
174.	$D(b \div a)(y\cdot b)$	Rp(59,156)		
175.	$D((b \div a)\cdot z)(y\cdot(b\cdot z))$	Rp(64,156)		
176.	$Dz((b \div a)\cdot z)$	Rp(64,156)		Sub(165)
177.	$Db(y\cdot(c\cdot c)) \vee [a=0]$	Rp(148,156)		

In	Clause	Source	Out	Disposition
178.	Dx0	Rp(4,171)		Sub(154)
179.	Dx(y•0)	Rp(4,171)		CNR(*)
180.	D1(y∘w)	Rp(5,171)		Sub(46)
181.	Dx(λ•w)	Rp(5,171)		Sub(11)
182.	$Dx(y_1 \cdot (y_2 \cdot (x \cdot w)))$	Rp(6,171)		
183.	$D(x_1 \cdot x_2)(y \cdot (x_1 \cdot (x_2 \cdot w))$	Rp(6,171)		
184.	$Dx(y \cdot (x \cdot w_1 + x \cdot w_2))$	Rp(7,171)		CNR(+)
185.	Dx((x•w)•y)	Rp(9,171)		CNR(*)
186.	Dx(y•(w•x))	Rp(9,171)		Sub(156)
187.	Dc(b•b)	Rp(17,171)		Sub(172)
188.	Da(y•(b•b))	Rp(17,171)		Sub(191)
+189.	Da(y•b)	Rp(59,171)	36	
190.	D(b ÷ a)(b•z)	Rp(64,171)		
191.	Da(y•(b•z))	Rp(64,171)		
192.	Da0	Rp(4,189)		Sub(154)
193.	Dab	Rp(5,189)		Sub(21)
194.	Da(x•(y•b))	Rp(6,189)		
195.	Da(b•y)	Rp(9,189)		
+196.	Da(c•c) v [a=0]	Rp(148,189)	37	
197.	Da(c•c) v [a=0]	Rp(9,196)		
198.	[a•((c•c) ÷ a) = c•c] v [a=0]	Cut(196,12)		
199.	Dac v [a=0]	Cut(196,51)	38	
+200.	∿Dac v [a=1] v [a=0]	Cut(58,18)	40	
+201.	[a=1] v [a=0]	+Cut(199,200)	42	
+202.	PN1 v [a=0]	Rp(201,16)	44	
203.	[1•(c•c) =:b•b] v [a=0]	Rp(201,17)		CNR(+)
204.	D1(b•b) v [a=0]	Rp(201,50)		Sub(46)
205.	D1(y•z) v Day v Daz v [a=0]	Rp(201,51)		
206.	D1b v [a=0]	Rp(201,58)		Sub(46)
207.	[1•(b ÷ a) =:b] v [a=0]	Rp(201,59)		CNR(+)
208.	[a•(b ÷ 1) =:b] v [a=0]	Rp(201,59)		

In	Clause	Source	Out	Disposition
209.	$1\cdot((b \div a)\cdot z) = b\cdot z \vee [a=0]$	Rp(201,64)		CNR(+)
210.	$a\cdot((b \div 1)\cdot z) = b\cdot z \vee [a=0]$	Rp(201,64)		
211.	$\underline{[1\cdot x + b\cdot z \neq 0]} \vee [a=0]$ $\vee [x + (b \div a)\cdot z = 0]$	Rp(201,89)		CNR(+)
212.	$[c\cdot c + (b \div 1)\cdot(-b) =: 0] \vee [a=0]$	Rp(201,121)		
213.	$[c\cdot c + ((b \div 1)\cdot(-b) + z) = z] \vee [a=0]$	Rp(201,126)		
214.	$[(b \div 1)\cdot b = c\cdot c] \vee [a=0]$	Rp(201,148)		
215.	$D1(y\cdot b) \vee [a=0]$	Rp(201,189)		Sub(46)
216.	$D1(c\cdot c) \vee [a=0]$	Rp(201,196)		Sub(46)
217.	$D1c \vee [a=0]$	Rp(201,199)		Sub(46)
218.	$\underline{\sim D1c} \vee [a=1] \vee [a=0]$	Rp(201,200)		
+219.	$[a=0]$	Cut(202,15)	45	
+220.	PNO	Rp(219,16)	47	
221.	$[0\cdot(c\cdot c) = b\cdot b]$	Rp(219,17)		
222.	$D0(b\cdot b)$	Rp(219,50)		
223.	$\sim D0(y\cdot z) \vee Day \vee Daz$	Rp(219,51)		
224.	$D0b$	Rp(219,58)		
225.	$0\cdot(b \div a) = b$	Rp(219,59)		CNR(+)
226.	$a\cdot(b \div 0) = 0$	Rp(219,59)		
227.	$0\cdot((b \div a)\cdot z) =: b\cdot z$	Rp(219,64)		CNR(+)
228.	$a\cdot((b \div 0)\cdot z) =: b\cdot z$	Rp(219,64)		
229.	0	Cut(220,14)		

REFERENCES

1. Allen, J. and Luckham, D., "An Interactive Theorem-Proving Program", in Machine Intelligence, Vol. 5 (1970), pp. 321-336.

2. Amarel, S., "On Representations of Problems of Reasoning about Actions", in Machine Intelligence, Vol. 3, D. Michie, ed. (Edinburgh University Press, Edinburgh, 1968), pp. 131-170.

3. Anderson, R., "Completeness Results for E-Resolution", AFIPS Conference Proceedings 36, 1970 Spring Joint Computer Conference (Montvale, New Jersey: AFIPS Press), pp. 653-656.

4. Anderson, R. and Bledsoe, W., "A Linear Format for Resolution with Merging and a New Technique for Establishing Completeness", J. ACM 17, 525-534 (1970).

5. Andrews, P. B., "Resolution in Type Theory", J. Symbolic Logic 36, 414-432 (1971).

6. Bledsoe, W., "Splitting and Reduction Heuristics in Automatic Theorem Proving", Artificial Intelligence 2, 55-77 (1971).

7. Bledsoe, W., Boyer, R. S., and Henneman, W. H., "Computer Proofs of Limit Theorems", Second International Joint Conference on Artificial Intelligence (Portsmouth: Grosvener Press, 1971), pp. 586-600.

8. Bledsoe, W. and Bruel, P., "A Man-Machine Theorem Proving System", Adv. Papers Third International Conference on Artificial Intelligence (Stanford University Press, Stanford Calif. 1973).

9. Bobrow, D. G. and Raphael, B., "New Programming Languages for AI Research", SRI Artificial Intelligence Center Technical Note 82 (Stanford Research Institute, Menlo Park, Ca., 1973).

10. Brinch-Hansen, P., "Concurrent Pascal: A Programming Language for Operating System Design", Information Science Technical Report No. 10 (California Institute of Technology, Pasadena, 1974).

11. Bruce, B. C., "A Model for Temporal References and Its Application in a Question Answering Program", Artificial Intelligence 3, 1-26 (Spring 1972).

12. Chang, C. L. and Lee, R. C., Symbolic Logic and Mechanical Theorem Proving (Academic Press, New York, 1973).

13. Cheatham, T. E. and Wegbreit, B. "A laboratory for the study of automatic programming", AFIPS Conference Proceedings 40, 1972 Spring Joint Computer Conference (AFIPS Press, Montvale, New Jersey) pp. 11-22.

14. Church, A., The Calculi of Lambda-Conversion (Princeton University Press, Princeton, N.Y., 1941).

15. Curry, H. B., Foundations of Mathematical Logic (McGraw-Hill Book Company, New York, 1963), Chap. 2.

16. Curry, H. and Feys, R., Combinatory Logic, Vol. I (Amsterdam: North-Holland Publishing Company, 1968).

17. Davis, M. and Putnam, H., "A Computing Procedure for Quantification Theory", J. ACM 7, 201-215 (1960).

18. Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R., Structured Programming (Academic Press, New York, 1972).

19. Enderton, H. W., A Mathematical Introduction to Logic (Academic Press, New York, 1972).

20. Fikes, R. E. and Nilsson, N. J., "STRIPS: A New Approach to Problem Solving", Artificial Intelligence 2, 189-208 (1971).

21. Gould, W. E., "A Matching Procedure for Omega-Order Logic", A. F. Cambridge Research Laboratories, Report AFCRL-66-781 (1966).

22. Green, C., "The Application of Theorem Proving to Question-Answering Systems", Ph.D. dissertation, Stanford Artificial Intelligence Project Memo AI-96 (1969).

23. Green, C., "Application of Theorem Proving to Problem Solving", Proc. International Joint Conference on Artificial Intelligence (1969), pp. 219-240.

24. Hayes, P. J., "A Logic of Actions", Machine Intelligence 6, ed. by B. Meltzer and D. Michie (American Elsevier Publishing Company, Inc., New York, 1971), pp. 495-520.

25. Hendrix, G., "Modeling Simultaneous Actions and Continuous Processes", Artificial Intelligence 4, 145-180 (1973).

26. Henkin, L., "Completeness in the Theory of Types", J. Symbolic Logic 15, 81-91 (1959).

27. Henkin, L., "A Theory of Propositional Types", Fundamentae Mathematecae 52, 323-344 (1963).

28. Herbrand, J., "Investigations in Proof Theory", in From Frege to Godel: A Sourcebook in Mathematical Logic, ed. by Van Heijenoort, (Harvard University Press, Cambridge, 1967), pp. 525-581.

29. Hewitt, C., "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", Ph.D. Thesis, Dept. of Mathematics, MIT, Cambridge, Mass., 1972.

30. Hindley, J. R., Lercher, B., and Seldin, J. P., Introduction to Combinatory Logic (Cambridge University Press, Cambridge, England, 1972).

31. Hoare, C. A. R., "An Axiomatic Definition of the Programming Language PASCAL", in International Symposium on Theoretical Programming, ed. by Andrai Ershov and V. A. Nepomniaschy (Springer-Verlag, New York, 1974), pp. 1-16.

32. Hoare, C. A. R., "Notes on Data Structuring", in [18].

33. Huet, G. P., "A Unification Algorithm for Type Theory", IRIA Laboria (1973).

34. Huet, G. P., "A Mechanization of Type Theory", in Adv. Papers 3d Intl. Conf. on Artificial Intelligence (Stanford University Press Stanford, Ca., 1973).

35. Ingargiola, G. P., "Hierarchies and Relations among Data Types", ACM Conference Proceedings (Fall 1974, San Diego), pp. 622-634.

36. Ingargiola, G. P., "Basic Features of CALEX", Progress Report No. 9 (California Institute of Technology, Pasadena, 1974).

37. King, J. C. and Floyd, R. W., "An Interpretation-Oriented Theorem Prover over Integers", J. Computer and System Soiences 6, 305-323 (1972).

38. Knuth, D. E., The Art of Computer Programming, Vol. 1: Fundamental Algorithms (Addison-Wesley, Menlo Park, California, 1968).

39. Knuth, D. E. and Bendix,P. B., "Simple Word Problems in Universal Algebra", Computational Problems in Abstract Algebra, ed. by John Leech (Pergamon Press, New York, 1970), pp. 263-298.

40. Kowalski, R., "Search Strategies for Theorem Proving", Machine Intelligence 5, ed. by B. Meltzer and D. Michie (American Elsevier Publishing Company, Inc., 1970), pp. 181-202.

41. Kowalski, R., "The Case for Using Equality Axioms in Automatic Demonstration", Symposium on Automatic Demonstration (Springer-Verlag, New York, 1968), pp. 112-127.

42. Kowalski, R. and Hayes, P. J., "Semantic Trees in Automatic Theorem Proving", Machine Intelligence 4, ed. by B. Meltzer and D. Michie (American Elsevier Publishing Company, Inc., 1969), pp. 87-102.

43. Kowalski, R. and Kuehner, D., "Linear Resolution with Selection Function", Artificial Intelligence 2, 227-260 (1971).

44. Kripke, S., "Semantical Considerations on Modal Logic", Acta Philosophica Fennica 16, 83-94 (1963).

45. Kuehner, D., "Some Special Purpose Resolution Systems", Machine Intelligence 7, ed. by B. Meltzer and D. Michie (American Elsevier Publishing Company, Inc., 1972), pp. 117-128.

46. Loveland, D., "A linear format for resolution", Symposium on Automatic Demonstration (Springer-Verlag, New York, 1970), pp. 147-162.

47. Luckham, D., "The resolution principle in theorem proving", Machine Intelligence 1 (American Elsevier Publishing Company, Inc., New York, 1967), pp. 47-61.

48. Luckham, D., "Refinement theorems in resolution theory", Symposium on Automatic Demonstration (Springer-Verlag, New York, 1970), pp. 163-190.

49. Luckham, D., "Some tree-paring strategies for theorem-proving", Machine Intelligence 3 (American Elsevier Publishing Company, New York, 1968), pp. 95-112.

50. McCarthy, J. and Hayes, P.J., "Some philosophical problems from the standpoint of artificial intelligence", Machine Intelligence 4, ed. by B. Meltzer and D. Michie (American Elsevier Publishing Company, Inc., New York, 1969), pp. 463-505.

51. Meltzer, B., "Prolegomena to a Theory of Efficiency in Proof Procedures," in Artificial Intelligence and Heuristic Programming, ed. by N. V. Findler and B. Meltzer (American Elsevier, New York, 1971), pp. 16-33.

52. Morales, Jorge J. (Unpublished research memorandum: Solution of group theory axiomatization problems using the interactive theorem prover) (Stanford, August 2, 1972).

53. Moses, Joel, "Algebraic simplification: A guide for the perplexed", C. ACM 14, 527-537 (1971).

54. Nilsson, Nils J., Problem Solving Methods in Artificial Intelligence (McGraw-Hill Book Company, New York, 1971).

55. _____,"Artificial Intelligence", AI Center Technical Note 89, 1974.

56. Norton, L. M., "Experiments with a heuristic theorem-proving program for predicate calculus with equality", Artificial Intelligence 2, 261-284 (1971).

57. Plotkin, G. D., "Building-in Equational Theories", Machine Intelligence 7, 73-90 (1972).

58. Pohl, Ira,"Heuristic Search Viewed as Path Finding in a Graph" Artificial Intelligence 1, 193-204 (1970).

59. Prawitz, D., "Advances and problems in mechanical proof procedures", Machine Intelligence 4, ed. by D. Michie and B. Meltzer (American Elsevier Publishing Company, New York, 1969), p. 59-72.

60. Richter, M. M., A Note on Paramodulation and the Functional Reflexive Axioms (University of Texas at Austin, Nov. 22, 1974; to be published.)

61. Robinson, G. A., Wos, L. T. and Carson, D. F., "Some Theorem-Proving Strategies and Their Implementations", Argonne National Laboratories, Technical Memo No. 72, 1964.

62. Robinson, G. A. and Wos, L. T., "Paramodulation and Theorem-Proving in First-Order Theories with Equality", Machine Intelligence 4, ed. by D. Michie and B. Meltzer (American Elsevier Publishing Company, New York, 1969), pp. 183-205.

63. Robinson, J. A., "A Machine-Oriented Logic Based on the Resolution Principle", J. ACM 12, 23-41 (1965).

64. Robinson, J. A., "Automatic Deduction with Hyper-Resolution", International Journal of Computer Mathematics 1, 227-234 (1965).

65. Robinson, J. A., "The Generalized Resolution Principle", Machine Intelligence 3, ed. by D. Michie and B. Meltzer (American Elsevier Publishing Company, New York, 1968).

66. Robinson, J. A., "Heuristic and Complete Processes in the Mechanization of Theorem Proving", Systems and Computer Science, ed. by J. F. Hart and S. Takasu (University of Toronto Press, Toronto, 1967).

67. Robinson, J. A., "An Overview of the Mechanical Theorem Proving", Theoretical Approaches to Non-Numerical Problem Solving, ed. by R. Banergi and M. D. Mesarovic (Springer-Verlag, New York, 1970), p. 2-20.

68. Robinson, J. A., "A Note on Mechanizing Higher Order Logic", Machine Intelligence 5, ed. by D. Michie and B. Meltzer (American Elsevier Publishing Company, New York, 1970), pp. 121-134.

69. Robinson, J. A., "Computational Logic: the Unification Algorithm", Machine Intelligence 6, ed. by D. Michie and B. Meltzer (American Elsevier Publishing Company, New York, 1971), pp. 63-72.

70. Schütte, K., "On Simple Type Theory with Extensionality", Logic, Methodology, and Philosophy of Science III, ed. by B. Van Rootselaar and J. F. Staal (North Holland Publishing Company, Amsterdam, 1968).

71. Shoenfield, J. R., Mathematical Logic, (Addison Wesley, Menlo Park, Calif., 1967)

72. Simon, H. A., "The Theory of Problem Solving", Information Processing 71, Vol. I, pp. 261-277.

73. Slagle, J., "Automatic Theorem Proving with Renamable and Semantic Resolution", J. ACM 14, 687-697 (1967).

74. Slagle, J., "Heuristic Search Programs", Theoretical Approaches to Non-Numerical Problem Solving (Springer-Verlag, New York, 1970), pp. 246-273.

75. Slagle, J., "Automatic Theorm Proving with Built-In Theories Including Equality, Partial Ordering, and Sets", J. ACM 19, 120-135 (1972).

76. Slagle, J., "An Approach to Finding C-Linear Complete Inference Systems", J. ACM 19, 496-516 (1972).

77. Waldinger, R. S. and Lee, R.C.T., "PROW: A Step toward Automatic Program Writing", Proc. International Joint Conference on Artificial Intelligence (1969), pp. 219-240.

78. Waldinger, Richard J., "Constructing Programs Automatically Using Theorem Proving", Ph.D. Dissertation (Carnegie-Mellon University, Pittsburgh, 1969).

79. Wos, L., Robinson, G. A. and Carson, D. F., "Efficiency and Completeness of the Set of Support Strategy in Theorem Proving", J. ACM 12, 536-541 (1965).

80. Wos, L., Carson, D. and Robinson, G., "The Unit Preference Strategy in Theorem Proving", Proc. AFIPS 1964 FJCC 26, Pt. II, (Spartan Books, 1964), pp. 615-621.

81. Wos, L., Robinson, G. A., Carson, D. V. and Shalla, L., "The Concept of Demodulation in Theorem Proving", J. ACM 14, 698-709 (1967).

82. Winograd, Terry, Understanding Natural Language (Academic Press, New York, 1972).

83. Wegbreit, B., "Heuristic Methods for Mechanically Deriving Inductive Assertions", Third International Joint Conference on Artificial Intelligence, Advance Papers (SRI Publications Department, Menlo Park, Calif., 1973).

84. Winston, P. H., New Progress in Artificial Intelligence, Annual Progress Report AI-TR-310, (MIT, Cambridge, 1974).