Addenda

The following sections correct, clarify and add some further details to issues covered in the thesis. These comments have been added in response to queries and suggestions offered by anonymous referees.

1 Errata

p 40: The heading "elimination elimination" should read "elimination rules". Also, the symbols Π_{I} and Π_{E} that occur in this figure stand for introduction proof components and elimination proof components respectively.

p 109: In the second paragraph the word "arhitecture" should be "architecture".

2 Notes

Whenever the term *axiom* is used, it should be understood as shorthand for the term *non-logical axiom.* Since natural deduction systems do not employ any *logical axioms*, this shorthand is not ambiguous.

It has been pointed out that all the examples in the thesis are in the datalog (function symbol free) form. This is to make the examples simple to understand. The natuaral deduction framework has nothing to say about the structure of terms. The reader should assume that we are dealing with first order terms, as defined in the beginning of chapter 2, throughout.

The proof of lemma 1 in chapter 2 supplies just the reduction steps needed for normalization. A proof of termination of the normalization procedure can be found in [Prawitz 65].

With reference to the positive definite language. described in chapter 4: The *dependency constraint* associated with the $\forall$I rule can be more efficiently implemented if we employ Skolem functions $f^S(X_1, \ldots, X_n)$ instead of just Skolem constants X^S. The parameters $X_1, \ldots, X_n$ being the set of free variables that occur in assumptions on

which the skolemized formula depends. N-Prolog [Gabbay & Reyle 84] and fragments of λ-Prolog [Miller et al. 89] offer well advanced implementations of this language.

The notion of *atomic normal form proof* is a rediscovery of *expanded normal form proof* published in [Prawitz 71]. That paper contains also an interesting discussion of the strengths of natural deduction proof theories compared to other approaches.

3 Sequent Calculus

An outline of the relationship between sequent calculus [Gentzen 35] and the natural deduction systems introduced in the thesis is interesting. Note that sequent calculus was historically derived from natural deduction.

In sequent calculus inferences no longer carry us from a collection of premiss formulae to a conclusion formula, but from a collection of premiss sequents to a conclusion sequent. A sequent being an object having the structure

$$F_1, F_2, \ldots, F_m \Rightarrow G_1, G_2, \ldots, G_n$$

where the Fs and Gs are formulae. The Fs here correspond to the *assertion formulae* and the Gs to the *goal formulae* of our natural deduction framework.

For classical predicate calculus the above sequent is informally equivalent to the formula

$$(F_1 \wedge F_2 \wedge \ldots \wedge F_m) \supset (G_1 \vee G_2 \vee \ldots \vee G_n)$$

Formally, the structure of the sequent is given meaning by the so called *structural rules of inference.* The CUT rule of inference introduced in the thesis is a natural deduction analog of the structural rule known as *cut* (*schnitt*). In sequent calculus we could write the CUT rule of the system $\mathcal{C}_\Sigma$ of figure 3.13 in this way

$$\frac{\Delta \Rightarrow B \qquad A, \Delta \Rightarrow C}{(\Delta \Rightarrow C)\Theta} \text{ where: } A\Theta = B\Theta$$

This analogy cannot however be taken too far. The thesis puts forward the idea that one should think of an asserted formula as a license to perform certain inferences. For example, a formula $A \wedge B \supset C$ has the force of the inference rule

$$\frac{\Delta \Rightarrow A \qquad \Delta \Rightarrow B}{\Delta \Rightarrow C} \text{ where: } (A \wedge B \supset C) \in \Delta$$

The function of the CUT rule is to build proofs using these derived rules or proof components. Still, both the sequent calculus cut and CUT mean transitivity of derivability, and hence the name.

4 Implementation

Comparisons were made between a Prawitz normal form and an atomic normal form minimal logic implementation for propositional examples (kindly supplied by Neil Tennant). These experiments were inconclusive about the comparative efficiency of atomic and non-atomic systems. For the example set given, differences in performance were small and favoured one system or the other depending on the problem.

Implementations, in the form of Prolog meta-interpreters, were developed for first order systems up to minimal logic (according to the classification of chapter 4). Work is continuing on more serious implementations, refer [Keronen 93].

References

[Gabbay & Reyle 84]

D. M. Gabbay and U. Reyle. *N-Prolog: An Extension of Prolog with Hypothetical Implications.* Journal of Logic Programming 4 319-355, 1984.

[Gentzen 35]

Gerhard Gentzen. *Untersuchungen über das logische Schliessen.* Mathematische Zeitschrift 39 176-210, 405-431, 1935. English translation in M. E. Szabo (ed). *The Collected Papers of Gerhard Gentzen.* North Holland, 1969.

[Keronen 93]

Seppo Keronen. *Non-Procedural Logic Programming.* Extensions of Logic Programming, Fourth International Workshop, Springer Lecture Notes in Artificial Intelligence (to appear).

[Miller et al. 89]

Dale Miller, Gopalan Nadathur, Frank Pfenning and Andre Scedrov. *Uniform Proofs as a Foundation for Logic Programming.* Report MS-CIS-89-36 LINC LAB 154, Department of Computer and Information Science, University of Pennsylvania, 1989.

[Prawitz 65]

Dag Prawitz. *Natural Deduction (A Proof-Theoretical Study).* Almqvist & Wiksell, 1965.

[Prawitz 71]

Dag Prawitz. 'Ideas and Results in Proof Theory', in J.E. Fenstad (ed). *Proceedings of the Second Scandinavian Logic Symposium.* North Holland, 1971.

Computational Natural Deduction

Seppo R. Keronen

A thesis submitted for
the degree of Doctor of Philosophy
of the Australian National University

November 1991

I hereby state that this thesis contains only my own original work, except where explicit reference is made to the work of others.

Seppo Keronen

Dedicated to
my mother
Leila Keronen
and my father
Reino Keronen

Abstract

The formalization of the notion of a logically sound argument as a natural deduction proof offers the prospect of a computer program capable of constructing such arguments for conclusions of interest. We present a constructive definition for a new subclass of natural deduction proofs, called atomic normal form (ANF) proofs. A natural deduction proof is readily understood as an argument leading from a set of premisses, by way of simple principles of reasoning, to the conclusion of interest. ANF extends this explanative power of natural deduction. The very detailed steps of the argument are replaced by derived rules of inference, each of which is justified by a particular input formula. ANF constitutes a proof theoretically well motivated normal form for natural deduction. Computational techniques developed for resolution refutation based systems are directly applicable to the task of constructing ANF proofs.

We analyse a range of languages in this framework, extending from the simple Horn language to the full classical calculus. This analysis is applied to provide a natural deduction based account for existing logic programming languages, and to extend current logic programming implementation techniques towards more expressive languages. We consider the visualization of proofs, failure demonstrations, search spaces and the proof search process. Such visualization can be used for the purposes of explanation and to gain an understanding of the proof search process. We propose an introspection based architecture for problem solvers based on natural deduction. The architecture offers a logic based meta language to overcome the combinatorial and other practical problems faced by the problem solver.

Keywords: computational logic, natural deduction, logic programming.

Contents

Preface

A central challenge for the discipline of deductive logic is to precisely characterize what is to count as an acceptable or *sound* argument in any domain of discourse whatsoever. In summary, an argument consists of a collection of statements (*premisses*) from which another statement (*conclusion*) is claimed to follow. Originally conceived to formalize reasoning in the domain of mathematics [Whitehead & Russell 1910-1913], classical first order predicate calculus provides, to date, the most widely accepted approach to meet this challenge.

A *deduction system* provides a formal, inductive definition of an argument. We will refer to this formal counterpart of an argument as a *proof*. Such a system offers the exciting prospect of a computer program which, given the description of a particular problem domain as premisses, is capable of constructing sound arguments for conclusions of interest. After early implementations revealed the computational intractability of this task, two important advances were made:

- The work of Alan Robinson [Robinson 65] on resolution refutation demonstrated a dramatic improvement in efficiency. The search space generated by the resolution rule of inference is smaller than for many other deduction systems. The fundamental unit of computation for the inference engine is the unification of two atomic formulae. The resolution refutation regime is readily applied in conjunction with goal directed and heuristic search strategies.

- Patrick Hayes [Hayes 73] and Robert Kowalski [Kowalski 79[a]] pioneered the idea that the search for solutions to deduction problems constitutes a worthy problem domain in itself. That is, a deduction problem consists of two distinct components, one contains knowledge describing the problem domain, the other contains knowledge to direct the search process.

The power of the above two ideas is captured beautifully in the logic programming language Prolog [Clocksin & Mellish 81]. We adopt the Prolog language, and its impressive implementation technology, as a point of reference for our investigation. We

propose a new approach to controlled deduction, still relying on essentially these same foundations, but incorporating the following two ideas:

- The natural deduction systems of Gerhard Gentzen [Gentzen 35] and Dag Prawitz [Prawitz 65] aim to reflect in a formal system rigorous arguments constructed by humans. Due to this similarity between natural deduction proofs and informal arguments, natural deduction is the standard proof theory for interactive proof editors, such as LCF [Paulson 87]. We argue that on these same grounds natural deduction is the right proof theoretic framework when explanation, debugging and control of deduction is required.

- Pick up any textbook on proof theory and you will notice that most of the discussion is at the meta level. The vocabulary is about formulae and proofs. Statements are schematic, meta-variables of various sorts standing for classes of these objects. We wish to offer this expressive power to the user of computer based problem solving systems. In particular, the search knowledge component of a deduction problem is best expressed as assertions in such a meta language. This idea has its roots in the artificial intelligence community [Hayes 73], [Davis76] and [Weybrauch 80].

While much of our investigation focuses on the application of these ideas to logic programming, it is also relevant for other applications. Computational logic is an important, unifying theme for research in artificial intelligence [Genesereth & Nilsson 87] and deductive databases [Minker 88]. The need for more expressive, yet computationally tractable languages is acutely felt in these areas [McCarthy & Hayes 69].

In addition to the citations already made, the following have had a major influence on the current work: Nils Nilsson [Nilsson 80] on rule-based deduction systems, Jon Doyle [Doyle 79] and Johan de Kleer [de Kleer 86] on truth maintenance, Seif Haridi [Haridi 81] on natural deduction based logic programming, Peter Schroeder-Heister [Schroeder-Heister 84] on extensions of natural deduction and Neil Tennant [Tennant 87] on relevant deduction.

I wish to thank Professor Robin Stanton for the opportunity to carry out the research reported here. His encouragement and sharp insights have been invaluable. Andy Bollen, Rob Edmondson, Kerry Taylor, Neil Tennant and Graham Williams also contributed a great deal. I am pleased to acknowledge financial support by the Australian National University and the Fujitsu Corporation. Thank you to my wife Salad and our son Sol for your patience and love.