

Annual Report

Computational Logic

Semi-Automated Meta-Theory Proofs for Sequent Calculi
Peter Chapman

Cut Admissibility is an important property for a calculus to have, and as such we wish to be able to automate the proof so that it can be fully formalised. The focus of this research is not to automate a given proof of Cut Admissibility, but rather to provide *proof templates*, which can then be adapted for other systems. However, before we can further expand on these aims, some background material must be presented. This material falls neatly into two categories; the basic theory, and the systems, and efforts, used to implement the theory.

1 Literature Review

1.1 Basic Theory

The main foundational text in this area is *Basic Proof Theory* [20]. It serves as a very good introduction to many topics, but most importantly the sequent calculus of Gentzen. Sequent calculi were introduced to be a more amenable system to meta-mathematical treatment than natural deduction. Whilst the form of sequents has changed from Gentzen's original paper, the spirit of them has not. A *sequent* is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are considered as multisets of formulae. Γ is known as the *context*. A standard interpretation is that if conjunction of formulae in Γ , the *antecedent*, is true then disjunction of formulae in Δ , the *succedent*, is true. A rule in this system is presented as in natural deduction, with a number of premisses being used to obtain a conclusion. For instance, the following is known as the right conjunction rule

$$\frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B} R\wedge$$

which tells us that if $\Gamma \Rightarrow A$ and $\Gamma \Rightarrow B$ are sequents, then so is $\Gamma \Rightarrow A \wedge B$.

Various different logical systems can be implemented in such a framework by careful restrictions of the sequents, and also the allowable rules. Intuitionistic logic, for example, usually requires that there is only a single formula in the succedent at all times, although this is not necessary. Multi-succedent intuitionistic calculi are allowable, and are an interesting area of study [10]. Likewise tableau calculi have been studied to circumvent certain problems with having a single formula in the succedent [19]. Troelstra and Schwichtenberg present many calculi in classical, intuitionistic and minimal formulations, and provide important results about them. Lots of work goes into proving cut admissibility, which can be seen as one of the central theorems of the book. Their method of constructively showing how cut can be eliminated from a sequent calculus deduction is therefore carefully expounded. It relies on a triple structural induction, or rather, induction on an ordered triple. As such, it uses the method of Dershowitz and Manna in [6]. Informally speaking, the measure that they created says that, given some ordering on the elements of a multiset, so long as there is a reduction in the number of elements of greatest magnitude, eventually the multiset will be empty.

A problem with some of the calculi given in *Basic Proof Theory*, most notably one called **G3i**, is that, during proof search, the rule for implication on the left can lead to an infinite branch in the proof tree, even though the propositional fragment of intuitionistic logic (**Ip**) is decidable. This is because the rule requires we copy the implicational formula being analysed into the antecedent of the first premiss:

$$\frac{\Gamma, A \supset B \Rightarrow A \quad \Gamma, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C} L \supset$$

In order to solve, in some way, this problem, Dyckhoff [7] and Hudelmaier [9] independently created the calculus **G4i**, which is the same as **G3i** except for the rules concerning implication on the left. They also proved the two systems had equivalent expressive power.

A more computer-friendly way to look at cut elimination is via terms and types. We can represent a deduction tree via a single term, albeit a complicated one. As an example, the rule for conjunction of the right given above could be displayed as

$$\frac{\Gamma \Rightarrow d_1 : A \quad \Gamma \Rightarrow d_2 : B}{\Gamma \Rightarrow \langle d_1, d_2 \rangle : A \wedge B}$$

where $h : C$ should be read as term h has type C . With a carefully formulated term notation, we can give only the final term and all the information

about the deduction will be contained within it. Removal of cut from a deduction can now be seen as a term rewriting system.

To study this further, the book of Baader and Nipkow [2] is useful. It does not explicitly refer to sequent calculi, rather general term systems, and describes the important properties, which are confluence and termination. *Confluence* of a term rewriting system means that if there are two different reductions from a term t , say $t \rightarrow t_1$ and $t \rightarrow t_2$, where \rightarrow is some kind of reduction step, then there exist two reduction paths $t_1 \rightarrow \dots \rightarrow w$ and $t_2 \rightarrow \dots \rightarrow w$ to a common term w . Given certain restrictions on \rightarrow , this is commonly known as the Church-Rosser theorem. *Termination*, on the other hand, says that *every* reduction path from a term t is finite. Cut elimination for certain systems can be shown to be terminating.

The book of Martin-Löf [12] is an introduction to the subject of *dependent types*; a way in which the value of a parameter affects its type. As an example, suppose we want to create a type for arrays of different sizes. We can use a type constructor that takes a natural number and returns a type, so the constructor could be something like `array(n)`. We say this constructor has *kind* $\text{Int} \Rightarrow \star$, where \star is the kind of all types. Quantification over dependent types is done using \forall and Π ; $\forall x^\tau. \sigma$ means that for every x of type τ , $\sigma(x)$ is a type, whereas $\Pi x^\phi. \underline{\kappa}$ means that for every x of type ϕ , $\underline{\kappa}(x)$ is a kind.

The idea of dependent types is used in the paper by McBride and McKinnin [14]. Here, a new framework is built for pattern matching. The fact that a value can refine a type is used as an extra level for computation. The standard example is to create a type family called **So** which is dependent on a boolean value, and then to only create types for **So true**. In this sense, when we see that a term p has type **So**($t_1 \vee t_2$) we get “for free” that $t_1 \vee t_2$ is true. This can be a useful tool when we come to characterise cut elimination, and the EPIGRAM system is built upon it. More of this will be described when we come to the implementation section.

Finally, the work of Ciabattoni and Terui [5] is similar work to what we propose, but from a different angle. They provide some conditions for a calculus that will ensure that cut is admissible in the calculus. This is very useful, however they do not give constructive arguments. So, whilst it is interesting to know for which calculi cut will be admissible, we wish to give a template for the transformation steps that will actually eliminate an instance of the cut rule.

1.2 Implementation

Formalisation of proofs of cut admissibility has been attempted before, in different systems. Pfenning [16] gives a formal proof in the TWELF system [17] which is almost identical to the proof given in [20]. It is formalised using the term notation which we explained earlier, and circumvents many of the problems usually encountered, most notably the representation of multisets. Both the intuitionistic and classical versions of the calculus **GK**, a variant of **G3** due to Kleene [11], are addressed. Whilst the term notation is easily dealt with by a computer, it is not the most easily readable by humans.

A more common system is ISABELLE [15]. This is a proof assistant which supports many different logics, and is extensible. The various tutorials and reference manuals are well written, and as such creating an implementation of **G3** is a relatively simple matter. There is another, more user-friendly and human readable layer built upon the system called ISABELLE/ISAR [21]. Proofs written in this layer can more easily be altered should the need arise, and they can also be structured in exactly the same way as an informal proof would be were we to write one out. Adams [1] noted in his thesis that ISABELLE was not adequately developed to assist in various meta-mathematical formalisations, in particular cut admissibility, due to the problems of defining new induction schemas. However, this work was done more than a decade ago, and new releases of ISABELLE have overcome this problem.

A paper by Ridge [18] of a related theorem to cut admissibility, known as Craig's Interpolation Theorem, shows just how useful ISABELLE can be. The Interpolation Theorem proceeds in a similar manner to cut admissibility, in that a number of cases and subcases are dealt with that correspond with the rules of the calculus. The representation of sequents that Ridge uses are easily readable by a human, and as such avoid some of the problems associated with terms. The specific form does have drawbacks, however, in that there is some redundancy which is eliminated when terms are used.

As stated earlier, the system EPIGRAM is relatively new [13]. As with most proof assistants, it is under continuous development. It is suited to proofs of cut admissibility because some of the case analysis is done automatically, or rather the creation of the cases is done automatically. A new release is planned, which should improve the usability of the system, in that more detailed error messages will be available, and some other, cosmetic changes will be made.

2 Research Plan

2.1 Work to Date

As said in the literature review, it is a relatively simple matter to define a sequent calculus in ISABELLE. We have implemented the calculi **G3cp** (the propositional fragment of classical logic), **G3ip** and **G4ip**, and also provided automatic proof tools to decide whether a given sequent is derivable in the calculus or not [3]. Whereas writing such a proof tool for **G3cp** is relatively straightforward, **G4ip** requires backtracking due to the non-invertibility of disjunction on the right, and **G3ip** requires some kind of loop-checking mechanism due to the left implication rule, as described earlier.

That work was done on proofs *within* the system. The focus of the work then became proofs *about* the system. These meta-mathematical proofs are results such as cut admissibility and the interpolation theorem. We have adapted Ridge’s proof of Craig’s interpolation theorem for the classical case to deal with the intuitionistic case [4]. The informal proof was completed in detail; in the literature [20] a sketch is given. Further to this the implementation was made applicable to the intuitionistic case, and then the proof script itself was altered.

2.2 Proposed Direction

The work on Craig’s Interpolation theorem was important. We expect to begin on implementing this proof in EPIGRAM. Then, we will formalise other meta-mathematical proofs, starting with the proof of cut admissibility for various systems, beginning with **G3i** and probably including **G4i** [8] and **G3c**. At this stage it will also become invaluable to survey the various proof assistants that could be used. Clearly ISABELLE/ISAR is an obvious candidate to begin with, but it is hoped that EPIGRAM may also be useful. It is these two which we will focus on to begin with.

The main aim of the project is to have machines help *produce* proofs of cut admissibility, rather than simply *check* proofs. We will look at the various proofs formalised in the chosen proof assistant, and attempt to generate some of the steps automatically. Whilst it would be ideal to generate the whole proof automatically, this may not be entirely feasible given the length of a PhD.

2.3 Milestones and their completion

We expect that the work on implementing the formalisation of Craig’s interpolation proof in EPIGRAM should take no more than two months. Following this, the work on cut admissibility should take a further three to five months. However problems could arise in this time. Most likely is that it would be quite difficult to implement cut admissibility proofs in EPIGRAM, and with a new release almost available some work could be rendered obsolete almost as soon as it is finished. In order to minimise this possible hindrance we plan to keep working in ISABELLE, which has a large user community and is relatively stable at the moment.

The informal proofs for cut admissibility all are very similar (for the three different calculi mentioned, and many others not mentioned). It is therefore hoped that the formal proofs will be likewise similar. If this is the case, then the main body of work should have a good chance of being in some way successful. As is usual, a problem at this stage will lead to a new direction for the project, which would be an investigation into why the work is not possible from a theoretical standpoint, or not possible at this time due to the proof assistants being unable to implement the work.

3 Other Activities

Throughout the year I have attended many GradSkills courses including *Introduction to Tutoring and Assesment*, *Project Management* and *How to get the best from your supervisor*, along with the compulsory post-graduate inductions. Recently I was present at the *Research Futures* conference. I have given two talks, one for the Computational Logic seminar series organised in school, and the other for a PhD reading party. I have been to seminars and workshops both at other universities and also in other Schools throughout the university. For instance, I attended a meeting of the Scottish Programming Languages Seminars (SPLS) in Glasgow, and a workshop on Proof Theoretic Semantics at St Andrews. I hope to go to a summer school for research students in Bertinoro in the summer, for which I have applied for funding.

During the second semester I tutored two classes in the module IS1001.

References

- [1] A. Adams. *Tools and Techniques for Machine-Assisted Meta-Theory*. PhD thesis, University of St. Andrews, 1997.
- [2] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.
- [3] P. Chapman. An automatic prover for sequent calculus in Isabelle. University of St Andrews Computer Science Research Report, available at www.dcs.st-andrews.ac.uk/~pc, 2007.
- [4] P. Chapman. Mechanising craig’s interpolation theorem for intuitionistic logic in Isabelle. University of St Andrews Computer Science Research Report, available at www.dcs.st-andrews.ac.uk/~pc, 2007.
- [5] A. Ciabattoni and K. Terui. Towards a semantic characterisation of cut-elimination. *Studia Logica*, 2006.
- [6] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 1979.
- [7] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 1992.
- [8] R. Dyckhoff, D. Kesner, and S. Lengrand. Strong cut-elimination systems for Hudelmaier’s depth-bounded sequent calculus for implicational logic. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR’06)*, volume 4130 of *LNAI*, pages 347–361. Springer-Verlag, August 2006.
- [9] J. Hudelmaier. Bounds for cut elimination in intuitionistic propositional logic. *Archive for Mathematical Logic*, 1992.
- [10] J. Hudelmaier. Semantische sequenzenkalküle, 1998. Habilitationsschrift, Fakultät für Informatik der Eberhard-Karls-Universität Tübingen, Germany.
- [11] S. C. Kleene. *Introduction to Metamathematics*. North-Holland Publishing Company, 1952.
- [12] P. Martin-Löf. *Intuitionistic Type Theory*. Number 1 in Studies in Proof Theory. Bibliopolis, 1984.

- [13] C. McBride. *Epigram: Practical Programming with dependent types*.
- [14] C. McBride and J. McKinna. The view from the left. *Journal of Functional Programming*, 2000.
- [15] T. Nipkow, L. Paulson, and M. Wenzel. *A Proof Assistant for Higher-Order Logic*. Number 2283 in Lecture Notes in Computer Science. Springer-Verlag, 2005.
- [16] F. Pfenning. Structural cut elimination i. intuitionistic and classical logic. *Information and Computation*, 2000.
- [17] F. Pfenning and C. Schuermann. *Twelf User's Guide*, 2005.
- [18] T. Ridge. Craig's interpolation theorem formalised and mechanised in Isabelle/HOL. *Logic in Computer Science*, 2006.
- [19] R. Smullyan. *First-order logic*. Springer-Verlag, 1968.
- [20] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Computer Science. Cambridge University Press, second edition, 2000.
- [21] M. Wenzel. *Isabelle/Isar Reference Manual*, 2002.