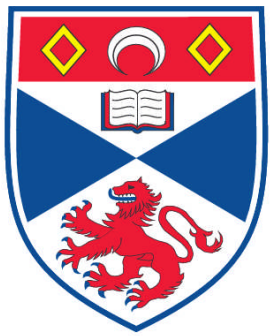
TOOLS AND TECHNIQUES FOR FORMALISING STRUCTURAL PROOF THEORY

Peter Chapman

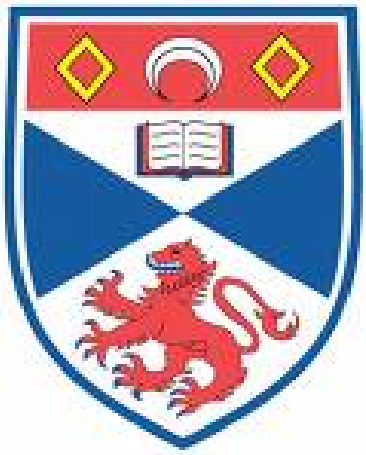
A Thesis Submitted for the Degree of PhD
at the
University of St. Andrews



2010

Tools and Techniques for Formalising Structural Proof Theory

Peter Chapman



A thesis submitted to the

University of St Andrews

for the degree of

Doctor of Philosophy

School of Computer Science, University of St Andrews

Abstract

Tools and Techniques for
Formalising Structural Proof Theory
Peter Chapman

Whilst results from Structural Proof Theory can be couched in many formalisms, it is the sequent calculus which is the most amenable of the formalisms to metamathematical treatment. Constructive syntactic proofs are filled with bureaucratic details; rarely are all cases of a proof completed in the literature. Two intermediate results can be used to drastically reduce the amount of effort needed in proofs of *Cut* admissibility: *Weakening* and Invertibility. Indeed, whereas there are proofs of *Cut* admissibility which do not use Invertibility, *Weakening* is almost always necessary. Use of these results simply shifts the bureaucracy, however; *Weakening* and Invertibility, whilst more easy to prove, are still not trivial. We give a framework under which sequent calculi can be codified and analysed, which then allows us to prove various results: for a calculus to admit *Weakening* and for a rule to be invertible in a calculus. For the latter, even though many calculi are investigated, the general condition is simple and easily verified. The results have been applied to **G3ip**, **G3cp**, **G3s**, **G3-LC** and **G4ip**.

Invertibility is important in another respect; that of proof-search. Should all rules in a calculus be invertible, then terminating root-first proof search gives a decision procedure for formulae without the need for back-tracking. To this end, we present some results about the manipulation of rule sets. It is shown that the transformations do not affect the expressiveness of the calculus, yet may render more rules invertible. These results can guide the design of efficient calculi.

When using interactive proof assistants, every case of a proof, however complex, must be addressed and proved before one can declare the result formalised. To do this in a human-readable way adds a further layer of complexity; most proof assistants give output which is only legible to a skilled user of that proof assistant. We give human-readable formalisations of *Cut* admissibility for **G3cp** and **G3ip**, *Contraction* admissibility for **G4ip** and Craig's Interpolation Theorem for **G3i** using the *Isar* vernacular of *Isabelle*. We also formalise the new invertibility results, in part using the package for reasoning about first-order languages, *Nominal Isabelle*. Examples are given showing the effectiveness of the formalisation. The formal proof of invertibility using the new methods is drastically shorter than the traditional, direct method.

Acknowledgements

I thank James McKinna (Radboud University, Nijmegen) for pointing out a gap in [Ridge, 2006]. This insight eventually resulted in [Chapman et al., 2008], which is covered in chapter 4. One of Jacob Howe's observations led me to the definition of combinable rules (Definition 22). Agata Ciabattoni (Technical University of Vienna) and I had a useful discussion about some of my early work on invertibility.

Jeremy Dawson (ANU, Canberra) made available to me some *Isabelle* files which he had written. The `extend` function from chapter §7 is owing to him.

The *Isabelle* group at TUM has been very supportive of my work. Christian Urban has in particular been fantastic: he was willing to help with problems which, sometimes, were entirely trivial. Without his help the thesis would not have come as far as it did. In particular, he showed me a better way of structuring proofs in *Isabelle*, thus contributing to the readability of [Chapman et al., 2008]. Makarius Wenzel, Tobias Nipkow and Stefan Berghofer have also been willing to help with my problems. Makarius gave me the formalisation of the lexicographic order used in a variety of places in this thesis, and Tobias gave me the insight to use type variables to index formulae and sequents in chapter 7. The *Isabelle* community at large answered my queries via the mailing list.

I thank Roy Dyckhoff for his efforts over the course of my PhD. His insights and critical eye have turned the work contained here into something worthy of a degree.

I thank my two examiners, Alex Simpson and Ian Gent, who spent the best part of a day picking over my thesis. Whilst it was long, it was never overly harrowing, and I thank them for that.

On a more personal note, my parents and family supported me throughout, and for this I will always be grateful. The St Andrews Links Trust allowed me to play borderline excessive amounts of golf, which was a great stress reliever. Both Matt and Natalie put up with me in the office for three years, and they were three very good years, with lots of laughs, and not many tears. Other friends and flatmates from my time in St Andrews are too numerous to mention. However, I cannot fail to thank Erica by name, since she suffered the lows and celebrated the highs with me.

Thank you.

Declaration

I, Peter Chapman, hereby certify that this thesis, which is approximately 40,000 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in September 2006 and as a candidate for the degree of Doctor of Philosophy in May 2007; the higher study for which this is a record was carried out in the University of St Andrews between 2006 and 2009.

Signature of candidate Date

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Signature of supervisor Date

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that the thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate the thesis into new electronic forms as required to ensure continued access to the thesis. We have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

Access to Printed copy and electronic publication of thesis through the University of St Andrews.

Signature of candidate Date

Signature of supervisor

Contents

Chapter 1

Introduction

1.1 Motivations

Mathematicians like proofs. More often than not, the more elegant a proof is, the more it is celebrated [Aigner and Ziegler, 2003]. Either way, though, the details of a proof should be available for inspection. Thus, the advent of computer checked proofs caused some unease; gone were the elegant proofs, but so too were the details. How could such proofs now be checked by a human? Proofs are evidence that a result is true; they give confidence. Does a result thrown back from a checker, based on several thousand lines of unreadable (to the casual reader) code, really give confidence? It should, yet it still leaves one feeling unsatisfied.

The counterpart to this is that mathematicians like shortcuts. Many text books are littered with proofs such as "Trivially", "Obviously", or "By inspection". For the diligent reader, filling in such gaps may take many hours of effort and many pages of writing. What may be trivial to the writers of a book may not be trivial to its readers. Indeed, this relegation of proofs to a word or two is really not much different from a computer checked proof: one cannot readily inspect the details. Yet, one is more acceptable than the other.

There are examples, of course, where a trivial proof is not really trivial at all. It may be similar to the proof of a similar theorem, but also subtly different, just different enough that it requires some thought. These lacunae may be overlooked for some time. A formally checked proof does not allow such omissions. Formalisation has found (and corrected) some gaps, such as a missing case in the proof of axiom expansion for a proof-search oriented formalisation of intuitionistic propositional logic, **G4ip** (the details are in section 4.5).

An area of mathematics where proofs are routinely omitted is structural proof theory. Many proofs use the same methods and are lengthy. It is rare to see all cases filled in with complete detail. It would seem, then, that structural proof theory is ripe for formalisation. But, the proofs can be elegant and furthermore give insight into how other similar proofs may go. So, what would be ideal would be a formally checked, but human readable, proof. Then,

the details could be inspected, and one could have more confidence in the result. There would be no gaps: all cases would have to be covered or the checker would not accept the proof. It is the search for human readable machine checked proofs, specifically of theorems from structural proof theory, which motivates this thesis.

1.2 Aims, Objectives and Contributions of the Thesis

Given the motivations the general objective of the thesis is straightforward: to provide examples, tools and techniques with which one can formalise structural proof theory in a human readable way. "Human readable" is a subjective phrase. A proof in a particular proof assistant may be fully readable to an expert user of that system, legible to a user of a similar system, and complete nonsense to the casual reader. Whilst it would not be feasible to create fully readable proofs for everyone, we at least hope to show that humanly legible proofs can be created.

The overriding aim is to show that formalisation is something which could, and perhaps should, be something which is undertaken at the same time as an informal proof is being drawn up. The closer one gets to formalised proofs looking identical to informal proofs, the closer one gets to being able to directly write the proof formally in the first place; one would not write an informal proof and then formalise it. A main contribution of this thesis is providing three examples of formalised proof theory. The results formalised are non-trivial, and furthermore the formalisations themselves are non-trivial.

Formalisation carries some burden, however. It may be quite obvious to the user that a certain result is true, but the system of choice steadfastly refuses to accept it. A further aim, then, is to make the formalisation process as pain-free as possible. The second main contribution of this thesis is to provide some theoretical results, which, when formalised, will reduce the burden on the user of the system. The results are interesting in their own right, but carry more weight when formalised, since then they can be applied to problems encountered elsewhere in the thesis.

1.3 Structure of the Thesis

This thesis is about formalising structural proof theory. Thus, before any detail is added a brief introduction is given to structural proof theory (chapter 2) and to formalisation (chapter 3). In the latter, the choice of interactive theorem prover for the thesis (*Isabelle*) is justified. Others were considered, but for one reason or another they were deemed deficient *for our purposes*. As development of the theorem provers is ongoing, had the project been started now, a different theorem prover might have been chosen. This is more fully explained in chapter 3.

We seek to combine chapter 2 and chapter 3 in chapter 4. Three non-trivial results from

structural proof theory are stated and proved in a form which, it is hoped, is readable to someone not well versed in *Isabelle*. Where necessary, we intersperse formal and informal proofs so that the similarities and differences can be appreciated. Two of the three formalised results fall victim to the same kind of bureaucracy: invertibility. We do not have recourse, in the formal proof, to the informal proof method "obviously." It is this which motivates chapters 5-7.

In chapter 5, we examine approaches to defining sequent calculi in a *general* fashion. Whilst all approaches have benefits, they are ultimately found lacking for our purposes, which motivates the definition of a new family of sequent calculi. With some tweaking of the criteria, a large range of calculi can be analysed. In chapter 6, several results are proved about these calculi, mainly about the invertibility of their rules. Having these informal results is interesting in itself, however the motivation for chapter 5 and chapter 6 was to cut the bureaucracy in *formal* proofs. Thus, in chapter 7, we formalise the invertibility results, and give concrete examples of how much time and effort one saves when using them.

What was achieved, relative to the stated aims of section 1.2, is discussed in chapter 8. This chapter also contains some possible further directions in which the work could be taken, along with a discussion of who could benefit from the material in this thesis.

Chapter 2

Structural Proof Theory

2.1 Introduction

Sequent calculi were invented by Gentzen as a tool to manipulate the formulae of a logic, much like his earlier formulation of natural deduction [Gentzen, 1969]. Unlike natural deduction, however, sequent calculi are very amenable to meta-mathematical analysis: it is easy to say when a sequent calculus derivation is, in some sense, normal or canonical. For this, one uses the notion of a cut-free derivation. The important question for a calculus then becomes: given a derivation involving cuts, is it possible to transform it into a derivation without cuts? In the process of answering this question, a number of other such questions arise about the structure of derivations. Thus, the area of Structural Proof Theory was created, and in this chapter, and indeed in the thesis as a whole, we will glimpse some of the power and elegance of sequent calculi.

2.2 Sequent Calculi

Definitions of what constitutes a *general* sequent calculus are hard to find. Given a logic, a *specific* sequent calculus is easy to define[1]. Most, but not all, are two-sided, the two sides of a sequent separated by some reserved symbol (throughout this thesis we will use the arrow $\Rightarrow$ to separate the left and right side, it is called the *sequent arrow*). Restrictions are placed on the structures of formulae which can appear on the left and right side of the sequent arrow: for (most) intuitionistic logics, a single formula can appear on the right-hand side, whereas in classical logic some more exotic structure can appear on the right. In [Gentzen, 1969], *sequences* of formulae were used on the left of the sequent arrow (and on the right in the calculus **LK**). Nowadays, it is more common to see sets or multisets of formulae as the chosen structure. The interpretation of a sequent is that the conjunction of the formulae on

[1] Whether it is easy to define a *sensible* or appropriate sequent calculus may not be so straightforward.

the left of the sequent arrow imply the disjunction of the formulae on the right of the arrow.

Some form of top-level or axiom sequents are permissible where the same (atomic) formula appears on the left and right of a sequent arrow, or we have $\bot$ on the left of the sequent arrow. From these initial sequents, derivations are built using a series of syntactic rules: it is possible to introduce connectives on the left or right of the sequent arrow. The root of any derivation is then provable in that calculus. As an example, consider the implicational fragment of intuitionistic logic with $\bot$ (henceforth known as **GImp**), which can be formulated by the following rules:

$$\frac{}{\Gamma, P \Rightarrow P}\ Ax \qquad \frac{}{\Gamma, \bot \Rightarrow C}\ L\bot$$

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

where P is some (meta-variable standing for a) propositional atom and Γ is a (meta-variable standing for a) multiset of formulae. The formula $A \supset B$ is propagated into the premisses of $L\supset$ because this rule is not invertible: without the copied formula, an application of the rule in proof search may not be sound. This calculus is derived from **G3ip**, a calculus for propositional intuitionistic logic (see appendix A).

Variations on the rule $L\supset$ create different calculi. The above rules form a *context-sharing* calculus: every multiple premiss rule has the same multiset of formulae on the left of each premiss. It is possible to create a *context-splitting* version of $L\supset$:

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

where each premiss has a (possibly) different context.

Of course, it is possible to build more exotic calculi for different purposes. Different sequent arrows can be used to differentiate between different judgements [Dyckhoff and Lengrand, 2006], distinguished formulae can appear as *stoups* above the sequent arrow [Girard, 1991], one-sided systems which dispense with the sequent arrow altogether [Schütte, 1977] and terms can be used [Dyckhoff and Pinto, 1999]. It is this flexibility which makes a general definition of sequent calculi difficult.

2.3 The Main Result

We have seen, briefly, what a sequent calculus looks like. But, for what are they useful? Gentzen developed them so that he could prove the completeness of classical and intuitionistic logic. There is one rule which is known as *Cut*, and is given in a variety of forms. Here, we show the context-sharing, single occurrence *Cut* rule [Troelstra and Schwichtenberg, 2000]:

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \text{ Cut}$$

If one can show that a derivation, in a calculus for classical or intuitionistic logic, with instances of *Cut* can be transformed into a derivation with no instances of *Cut* (a *cut-free* derivation) then one can deduce that the calculus, and hence the logic, is consistent. For, it becomes impossible to give a derivation with root:

$$\Rightarrow \bot$$

or, alternatively, derive a sequent with an empty right-hand side. One can see this by observing that none of the rules for a calculus for classical or intuitionistic logic allows an empty right-hand side.

Cut elimination (for a calculus with the rule *Cut*), or showing *Cut* admissibility (for a calculus without *Cut*) then becomes an important property for a calculus to have. Gentzen called this the *Hauptsatz* [Gentzen, 1969]. We will show *Cut* admissibility for the implicational fragment of intutionistic logic, which should highlight the main issues in larger proofs. Before that is possible, however, we need to show three other meta-theoretical properties hold for **GImp**: admissibility of *Weakening*, *Inversion* and *Contraction*.

Weakening allows one to add additional formulae to the context of a sequent [Troelstra and Schwichtenberg, 2000]. Furthermore, it is possible to do this in such a way that the height of the derivation of a sequent does not change. This result and proof, as well as the others in this chapter, are standard (see, for example, [Troelstra and Schwichtenberg, 2000] and [Dragalin, 1988]) and included to highlight some of the problems encountered.

The following definition from [Dyckhoff and Negri, 2000] and [Negri and von Plato, 2001], is extensively used in later chapters, is required for the theorems which follow:

Definition 1 (Height-Preserving Admissibility) *The rule R given by:*

$$\frac{S}{S'} \ R$$

*is **height-preserving admissible** in a calculus iff for every n and every derivation of height n of an instance of S there is a derivation of height $\leq n$ of the corresponding instance of S'.* $\dashv$

There is a corresponding definition of *admissibility*, which drops information about height. Height preserving admissibility therefore implies admissibility.

Theorem 1 (Weakening) *The rule:*

$$\frac{\Gamma \Rightarrow C}{\Gamma, A \Rightarrow C}\ w$$

is height-preserving admissible in ***GImp****.*

Proof. Induction on the height, n, of the derivation of the premiss. If $n = 0$, then the derivation is an axiom or an instance of $L\bot$. In either case, we can add an extra formula to the context and the conclusion will still be an axiom or an instance of $L\bot$, and hence will be derivable at height 0.

If $n > 0$, then the last rule used in the derivation of the premiss is either $R\supset$ or $L\supset$. In the former, we have $C \equiv D \supset E$, and the derivation ends with:

$$\frac{\Gamma, D \Rightarrow E}{\Gamma \Rightarrow D \supset E}$$

Using the induction hypothesis on the premiss of this, we obtain the following derivation:

$$\frac{\dfrac{\Gamma, D \Rightarrow E}{\Gamma, A, D \Rightarrow E}\ ih}{\Gamma, A \Rightarrow D \supset E}\ R\supset$$

If the last rule used is $L\supset$, then the derivation ends with:

$$\frac{\Gamma', D \supset E \Rightarrow D \quad \Gamma', E \Rightarrow C}{\Gamma', D \supset E \Rightarrow C}$$

Then, we use the induction hypothesis on both premisses, and then apply $L\supset$ again to complete the case and the proof. ⊣

At some stages in a derivation, we may wish to know what the premisses were for a given conclusion. In other words, we want to know which rules we can invert. Given a derivable sequent which matches the conclusion of a rule, when is it safe to say it could have been an instance of that rule which derived the sequent? This is another of the key results for a sequent calculus. If all rules are invertible, then properties such as *Cut* admissibility become much easier to show (as will be discussed in section 5.6). For **GImp**, the rule $R\supset$ is invertible. However, the rule $L\supset$ is only partially invertible: given a sequent matching the conclusion, we can only prove that the right premiss is derivable without an increase in height. In what follows, we say a formula is *principal* for a rule if it is introduced by that rule [Troelstra and Schwichtenberg, 2000]. Axioms are not counted as rules, thus $\bot$ is not counted as being principal in $L\bot$. A formula is principal for a derivation if it is principal for the last rule used in that derivation:

Theorem 2 (Invertibility) *The rule:*

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

is height-preserving admissible in ***GImp****.*

Proof. Induction on the height n of a derivation of the premiss. If $n = 0$, then the premiss is an axiom or an instance of $L\bot$. In the former case, the conclusion is likewise an axiom since we restrict to propositional atoms for axioms, so $A \supset B$ could not form part of an axiom (see section 5.7). In the latter case, the conclusion is also an instance of $L\bot$.

If $n > 0$, then the last rule used in the derivation is either an instance of $R\supset$ or an instance of $L\supset$. In the former, suppose $C \equiv D \supset E$. Then, the derivation ends with:

$$\frac{\Gamma, A \supset B, D \Rightarrow E}{\Gamma, A \supset B \Rightarrow D \supset E}$$

Applying the induction hypothesis to the premiss and then applying $R\supset$ yields the result.

When the last rule used is $L\supset$, there are two further subcases: one in which $A \supset B$ is principal and one in which there is some other formula $D \supset E$ which is principal. In the latter, we apply the induction hypothesis to each premiss, then apply $L\supset$, which gives the result. In the former, the derivation ends with:

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

so we simply take the right derivation, and we are done. ⊣

Another structural rule which some calculi admit is *Contraction*: if we have two copies of a formula in the context of a sequent, we can remove one. This rule is automatically admissible if one uses sets for contexts, since sets have no multiplicity. The following two theorems use a lexicographic order as the induction measure [Baader and Nipkow, 1999].

Theorem 3 (Contraction) *The rule:*

$$\frac{\Gamma, A, A \Rightarrow C}{\Gamma, A \Rightarrow C}\ \mathit{Contr}$$

is height-preserving admissible in ***GImp****.*

Proof. By induction on the complexity of the contracted formula A, with a subinduction on the height n of the derivation of the premiss.

When A is a propositional atom, say P, then either $C \equiv P$ or not. In the former case, the conclusion is likewise an axiom. In the latter case, if the derivation is by an axiom (for

some other propositional atom) or an instance of $L\bot$, then the conclusion is likewise an axiom or an instance of $L\bot$. If the derivation is by a logical rule, then apply the induction hypothesis in the premisses of that rule, then the rule again. For instance, say the last rule used in the derivation is $L\supset$ with principal formula $D \supset E$. Then, the following derivation suffices:

$$\dfrac{\dfrac{\Gamma, D\supset E, P, P \Rightarrow D}{\Gamma, D\supset E, P \Rightarrow D}\ ih \quad \dfrac{\Gamma, E, P, P \Rightarrow C}{\Gamma, E, P \Rightarrow C}\ ih}{\Gamma, D\supset E, P \Rightarrow C}\ L\supset$$

If the contracted formula is $\bot$, then the conclusion is an instance of $L\bot$.

If the contracted formula is $D \supset E$, then we perform case analysis on the last rule used in the derivation. If it was an axiom or an instance of $L\bot$, then the conclusion is likewise an axiom or instance of $L\bot$. If the last rule used is $R\supset$ or an instance of $L\supset$ where $D\supset E$ is non-principal, then we apply the induction hypothesis at the lower height of the premisses, and we are done.

This leaves the case where the last rule used is an instance of $L\supset$ where $D \supset E$ was principal:

$$\dfrac{\Gamma, D\supset E, D\supset E \Rightarrow D \quad \Gamma, D\supset E, E \Rightarrow C}{\Gamma, D\supset E, D\supset E \Rightarrow C}$$

Apply the (sub)induction hypothesis to the left premiss to contract $D \supset E$. To the right premiss apply theorem 2, contract the smaller formula E, then apply $L\supset$:

$$\dfrac{\dfrac{\Gamma, D\supset E, D\supset E \Rightarrow D}{\Gamma, D\supset E \Rightarrow D}\ ih \quad \dfrac{\dfrac{\Gamma, D\supset E, E \Rightarrow C}{\Gamma, E, E \Rightarrow C}\ L\supset_{inv}}{\Gamma, E \Rightarrow C}\ ih}{\Gamma, D\supset E \Rightarrow C}\ L\supset$$

⊣

Thus, after several straightforward proofs, we are ready to show the main result: *Cut* admissibility. The proof follows the format of that in [Troelstra and Schwichtenberg, 2000] for intuitionistic propositional logic.

Theorem 4 (Cut admissibility) *For the calculus* ***GImp*** *the Cut rule:*

$$\dfrac{\Gamma \Rightarrow A \quad A, \Gamma \Rightarrow C}{\Gamma \Rightarrow C}\ Cut$$

is admissible.

Proof. By induction on the complexity of the cut formula A, with a subinduction on

the lexicographic order of the heights (n, n') of the derivations of the premisses[2]. There are three cases to consider:

1. Both of the premisses used in the instance of *Cut* are axioms: Ax or $L\bot$, OR
2. At least one premiss is not an axiom and the cut formula A is not principal in at least one derivation used in the instance of *Cut*, OR
3. The cut formula A is principal in both derivations used in the instance of *Cut*.

Case 1. Again, we split into cases. If the left premiss or right premiss is an instance of $L\bot$, then the conclusion of the rule is likewise an instance of $L\bot$.

If the left premiss is an axiom, then there is some propositional atom $P \in \Gamma$. We inspect the right premiss: if it is an axiom for P, then the conclusion is likewise an axiom for P. If it is an axiom for some other propositional atom Q, then the conclusion is likewise an axiom for Q (since $Q \in \Gamma$).

Case 2. If A is not principal in the left derivation, then the last rule used must be $L\supset$. Then, the left derivation ends with:

$$\frac{\Gamma', D \supset E \Rightarrow D \quad \Gamma', E \Rightarrow A}{\Gamma', D \supset E \Rightarrow A}$$

whilst the right premiss of the instance of *Cut* can be rewritten to:

$$A, \Gamma', D \supset E \Rightarrow C$$

Invert this sequent using theorem 2, then use the induction hypothesis to cut A at a lower height then apply $L\supset$ again:

$$\dfrac{\Gamma', D \supset E \Rightarrow D \qquad \dfrac{E, \Gamma' \Rightarrow A \qquad \dfrac{A, \Gamma', D \supset E \Rightarrow C}{A, \Gamma', E \Rightarrow C}\, L\supset_{inv}}{E, \Gamma' \Rightarrow C}\, ih}{\Gamma', D \supset E \Rightarrow C}\, L\supset$$

Note it is possible to do this without using explicit inversion: weaken with E and $D \supset E$ in appropriate places, cut A, then finish with two copies of $D \supset E$ on the left. An application of theorem 3 then completes the case.

If A is not principal for the right premiss, then a similar argument gives the result, and so the proof is omitted.

Case 3. If the cut formula is principal for both premisses, then the derivation ends with:

[2]The sum of the heights of the premisses is not used, although it could be.

$$\dfrac{\dfrac{\Gamma, D \Rightarrow E}{\Gamma \Rightarrow D \supset E}\, R\supset \quad \dfrac{\Gamma, D \supset E \Rightarrow D \quad \Gamma, E \Rightarrow C}{\Gamma, D \supset E \Rightarrow C}\, L\supset}{\Gamma \Rightarrow C}$$

Then, using the induction hypothesis based on height to cut $D \supset E$, then twice more based on the complexity of the cut formula to cut D and E, we get the following derivation:

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow D \supset E \quad \Gamma, D \supset E \Rightarrow D}{\Gamma \Rightarrow D}\, ih \quad \Gamma, D \Rightarrow E}{\Gamma \Rightarrow E}\, ih \quad \Gamma, E \Rightarrow C}{\Gamma \Rightarrow C}\, ih$$

This completes the case, and the proof. ⊣

The above proof should be quite convincing. Indeed, it was written as an example of a complete proof, but careful subsequent analysis showed a case is not considered[3]. Consider the following table, where the rows correspond to the last rule used in the left premiss of the instance of *Cut* and the columns correspond to the last rule used in the right premiss of the instance of *Cut*. For three of the rules, the principal formula of the rule may or may not be principal in the instance of *Cut*: those where the principal formula of the rule is also the principal formula of the instance of *Cut* are marked "p", otherwise they are not principal for the instance of *Cut* and are marked "np". The column for $R\supset$ is necessarily non-principal. Some combinations cannot happen; these are marked with an "-". The possible cases are marked with a 1,2 or 3, which corresponds to the case in the proof above which covers it:

	Ax,*p*	*Ax*,*np*	$L\bot, p$	$L\bot, np$	$L\supset, p$	$L\supset, np$	$R\supset$
Ax	1	1	-	1	-	2	2
$L\bot$	1,2	1,2	1,2	1,2	**(?)**	2	2
$L\supset$	2	2	2	2	2	2	2
$R\supset$	-	2	-	2	3	2	2

The case where the last rule on the left is $L\bot$ and the last rule on the right is a principal use of $L\supset$ is not covered in the proof. It is simple to fix: $\bot$ will be in Γ so the conclusion is likewise an instance of $L\bot$. However, that it is overlooked quite easily is more important. Whilst this gap was discovered without formalisation, had the result been formalised it would have been immediately obvious that a case was missing.

The proof of theorem 4 only had a few cases, since there was only one connective one has to consider. The more connectives one has, the more cases where a formula is non-principal one must consider. For example, the calculus **G3ip** [Troelstra and Schwichtenberg, 2000]

[3]The omitted case in the proof was addressed: only the proof was not shown.

has three connectives, with 7 logical rules. This creates 62 cases where neither rule is an axiom. It would be an extravagant use of space to give full derivation transformations for each case. Indeed, Gentzen did not give full details for every case [Gentzen, 1969]. Normally, one would see such proofs given as "obviously" or "similarly." It should cause slight unease that all cases may not be fully inspected; it only takes one case to be slightly non-standard for a proof by such means as "obviously" or "similarly" to be rendered deficient. Moreover, the more connectives there are, the more likely it is that there will be some oversight like that outlined above. The pertinent questions are then:

- What short-cuts can be taken so that the result is fully proved without wading through many bureaucratic details?
- What can be done to gain more confidence in the result?

Both questions will be addressed in this thesis. It is the latter which motivates the next chapter.

2.3.1 Short-cuts to Cut admissibility

Various authors have shown sufficient conditions for *Cut* admissibility [Ciabattoni and Terui, 2006a],[Ciabattoni and Terui, 2006b], [Restall, 1999],[Curry, 1963]. The problem with them is that they are applicable to a small collection of calculi only: the conditions one must place upon such calculi are very restrictive. This will be addressed in more depth in chapter 5.

Chapter 3

Formalising Mathematics

3.1 Introduction

We saw in the previous chapter that many cases in Structural Proof Theory proofs are left, in some sense, unfinished. The theoretical approach would have us search for short-cuts which reduce the number of cases to be considered. The other route to gain more confidence in a result is through formalisation. A machine-checked proof will not allow hand-waving or glossing over of parts of a proof: all the details must be filled in. So long as the checker can be verified as correct and the formalisation is faithful, one can then have full confidence in the result.

A fully checked proof script also can be used as a pedagogical aid, either to the proof itself, or to how other such machine-checked proofs may proceed.

In this chapter, we examine some proof assistants and investigate some formalisations of Structural Proof Theory.

3.2 An Introduction to Epigram

Epigram is a dependently-typed system [McBride and McKinna, 2004], [McBride, 2005]. An introduction to dependent types can be found in [Martin-Löf, 1984]. Types are created in a natural deduction style: base constructors are introduced with no premisses, and inductive cases are introduced with at least one premiss. For example, consider the type of lists (of objects with type A) of specified length. The empty list has length zero and, given a list of length n and an object of type A, then adding this object to the list will create a list of type $n + 1$. Using the syntax of Epigram, this becomes (assuming the type *Nat* has been defined, as has the function +):

$$\underline{\text{data}} \quad \frac{n : \text{Nat} \quad A : \star}{\text{list } n\ A : \star} \quad \underline{\text{where}} \quad \frac{}{\text{nil } : \text{list } 0\ A} \ ; \ \frac{a : A \quad as : \text{list } n\ A}{\text{cons } a\ as : \text{list } (n+1)\ A}$$

To define a function on such a structure, one uses the key-word `let` followed by the type of the function, for given arguments. For instance, we can define the append function:

$$\underline{\text{let}} \quad \frac{xs : \text{list } n\ A \quad ys : \text{list } m\ A}{\text{append } xs\ ys : \text{list } (n+m)\ A}$$

The system responds by asking for a definition. One can use recursion on one of the arguments (with the keyword `rec`), case analysis on one of the arguments (with the keyword `case`) and so on, and then fill in the blanks. In Epigram the required definition is:

$$\begin{array}{l}\text{append } xs\ ys \Leftarrow \texttt{rec}\ \ xs\ \{\\ \quad \text{append } xs\ ys \Leftarrow \texttt{case}\ \ xs\ \{\\ \quad\quad \text{append nil } ys \Rightarrow ys\\ \quad\quad \text{append } (\text{cons } a\ as)\ ys \Rightarrow \text{cons } a\ (\text{append } as\ ys)\}\}\end{array}$$

Everything on the left-hand side of the arrows is generated by the system, the user inputs the right-hand side (including the arrows).

Using the methods of Epigram, one can encode sequents and derivations as being terms of a dependent type. For instance, to encode the height, one would index the type with a natural number. Then, various functions could be proved about them, which are automatically type-checked by the system (lines appear as green if they type check, yellow if more information is needed, and brown if they are false). However, the interface is very buggy [McBride, 2005] and thus is not suitable for large scale developments. Indeed, Epigram is now implemented in Agda [Norell, 2009], [Bove et al., 2009], a much more user-friendly environment. This development came too late for this thesis, but would be an interesting project to emulate the work contained here in a dependently typed language.

A new version of Epigram is being developed [McBride, 2008]. Some of the problems with Epigram will be cleared up, including some predefined types which one might like to use.

3.3 An Introduction to Coq

Coq [Bertot and Castéran, 2004], [Coq Development Team, 2006] is based on the Calculus of Constructions [Coquand and Huet, 1988] and the Calculus of Inductive Constructions [Pfenning and Paulin-Mohring, 1989] and, like Epigram, uses dependent types. However, it has much larger libraries, a larger user-community (to solve problems) and a much nicer interface, including its own integrated development environment, the *CoqIDE*. Thus, it is a much more useful tool for large scale theory development. For instance, the four colour theorem has been formalised in Coq, along with other important results[1].

Users can define datatypes, and then prove results about them using induction and a

[1] Details are available at http://ralyx.inria.fr/2004/Raweb/logical/uid40.html#uid40

variety of tactics. In this respect, it is not dissimilar to *Isabelle*, although the syntax is different. As such, a full introduction is not given here; the interested reader is directed to the various tutorials provided on the Coq web-page (http://coq.inria.fr/) and also to [Adams, 1997], where the use of Coq to formalise sequent calculi was investigated in depth. In [Herbelin and Lee, 2009], a model-based approach to *Cut* elimination was formalised in Coq. Since Coq is constructive, one can then extract a *Cut* elimination algorithm for intuitionistic sequent calculi.

One of the main deficiencies at the time of starting the project was that Coq only had a tactic driven approach to proof. In other words, one repeatedly manipulated the goal statement by applying induction, case analysis, simplification etc., until all that remained were trivial subgoals. Thus, one would end up with a proof tree in the natural deduction style. From this, one could extract a lambda calculus proof term. For the casual reader, both the proof term and the proof script which generated the term are not particularly edifying. The proof script in particular gives no insight into the internal proof state at any particular point. Thus, if one wishes to observe the methods used, the entire proof needs to be rerun step-by-step. One of the main aims of the work presented here is to simulate "book proofs" of results in a human-readable way. Obviously, then, this root-first approach is unsatisfactory. Moreover, reusing of proofs, or rather, adapting proofs to prove similar, related results, becomes a chore. Also, as often happens with software which is constantly undergoing development, tactic-style proofs may not be instantly verified by new versions of Coq. Such a proof will need to be updated by hand (although they should become simpler as the tactics become more powerful).

However, a declarative style of proof has been developed for Coq: C-zar [Corbineau, 2008]. It follows in the traditions (both stylistic and the rhyming sound) of Mizar [team, 2008] and *Isar* [Wenzel, 2002]. This was developed too late for this project.

The handling of variable binding and naming in Coq was also lacking when this project was begun. However, this has also been addressed in [Aydemir et al., 2007], which is similar to the Nominal techniques in *Isabelle* and [Tasson and Urban, 2005]. Thus, there is little, except perhaps the level of sophistication that *Isar* and *Nominal Isabelle* developments may[2] have over C-zar and the work of [Aydemir et al., 2007] respectively, to suggest that the work of this thesis could not be formalised in Coq rather than *Isabelle*. However, this was not the case when the project was started.

3.4 An Introduction to Isabelle

Isabelle is a proof assistant with a small collection of inbuilt tactics, a small trusted kernel and a large number of user supplied libraries. In this section, we will examine some of the main syntax, tools and methodologies of *Isabelle*, including the *Isar* vernacular [Wenzel,

[2] *Isar* and *Nominal Isabelle* have been around for some time, and thus under development for longer.

2002], [Wenzel, 2006]. *Isar* was created to give proofs which could be read independently of the *Isabelle* system. A pure *Isabelle* tactic script gives no information as to the internal proof state, and so is unreadable to a human. To discover how the proof works in detail, one must proceed through the proof step-by-step in *Isabelle*, which displays the current proof state. In *Isar*, we have the ability to explicitly name and display assumptions. This facility, along with the use of English words in a proof, means it is possible to infer the proof state at any time *without* running the proof through *Isabelle*. A more complete introduction can be found in [Nipkow et al., 2005], which is the tutorial which comes with the *Isabelle* distribution.

Isabelle allows for a lot of syntactic sugar. It also interfaces nicely with LaTeX to produce fully checked typeset documents. All *Isabelle* code which appears in this thesis has been verified, although the details are not always shown. Indeed, the LaTeXproof fragments can only be produced once the associated proof has been checked by the *Isabelle* system. The approach of *Isabelle* is akin to functional programming. Datatypes are defined by giving a number of constructors. For instance, consider the lists of natural numbers. This is defined as follows, where we have added the abbreviation "##" for readability:

datatype *natlist* = *NIL*
| *CONS nat natlist* (- ## -)

The notation $(-\#\#-)$ after the declaration for *CONS* tells the *Isabelle* system that this is the abbreviation we wish to use. In particular, the dashes tell the system where each of the variables appear in the abbreviation. Whilst this is not so important here, since there are only two variables, it becomes more useful for higher arity functions.

We can then define the functions `reverse` (not shown), `append` (abbreviated by "@@", not shown) and `sum`, which adds a list of natural numbers, by primitive recursion, which is introduced by **primrec**:

consts *sum* :: *natlist* ⇒ *nat*
primrec
sum NIL = *0*
sum (*x* ## *xs*) = *x* + (*sum xs*)

We can then show the simple lemma that the sum of a list, and the sum of a reversed list, are the same. In the proof, the lemma `sumappend` is used: this is also shown, albeit with a simple proof. This proof is a very short pure *Isabelle* proof. It first performs induction on the list `ns`, then finishes all the generated subgoals by the `auto` method of *Isabelle*. For the reverse sum proof, we use the declarative style *Isar*, which aims to give human-readable proofs which are nonetheless machine checked:

lemma *sumappend*:
shows *sum* (*ns* @@ *ms*) = *sum ns* + *sum ms*
by (*induct ns*) *auto*

```
lemma revsum:
  shows sum ns = sum (reverse ns)
proof (induct ns)
  case NIL
  have sum (reverse NIL) = sum NIL by auto
  then show sum NIL = sum (reverse NIL) by auto
next
  case (CONS m ms)
  then have IH: sum ms = sum (reverse ms) by simp
  moreover
      have sum (reverse (m ## ms)) = sum ((reverse ms) @@ (m ## NIL))
      by auto
  with sumappend[where ns=reverse ms and ms=m##NIL]
      have sum (reverse (m ## ms)) = sum (reverse ms) + m by auto
  ultimately
      have sum (reverse m ## ms) = m + sum ms by auto
  then show sum (m ## ms) = sum (reverse m ## ms) by auto
qed
```

The above proof demonstrates most of the keywords in an *Isar* proof. We use induction on the structure of *ns*, which then creates two cases. The **case** keyword then allows us to work on one of the two cases at a time. The word **have** followed by a statement creates a new subgoal for us to prove, which we can do using the **by** keyword, followed by some proof tactic which will prove that subgoal. Other important linking words are **then** and **with**. These use the statement on the previous line, or use the statement on the previous line along with the fact given respectively, in the proof of the new statement. Note that the only statements which are available to use on each line are those which are explicitly stated in that line, either by use of **then** and **with**. The keyword **show** lets us state and prove a top-level goal; here they allow us to finish the subcases of the inductive proof.

Most complicated is the **moreover-ultimately** pair. This joins facts together across separate blocks. So, when we type **ultimately**, we allow both the statement labelled *IH* and the statement:

$$\text{sum (reverse } m\#\#ms) = \text{sum (reverse } ms) + m$$

to be used in the upcoming proof step. One can use arbitrarily many **moreover** steps before an **ultimately** step.

We also see the reusing of earlier results. The lemma `sumappend` has been specialised using the **where** command: this instantiates the variables in the lemma.

There are various proof tactics one can use to prove subgoals. The method `simp` uses the *Isabelle* simplifier. It tries to apply a lemma from the library as a rewrite rule, and

returns the new proof state if the method does not prove the subgoal. The lemmas which it tries to apply are ordered in some way, but it is possible to force the simplifier to use specific lemmas using the keywords `add`, `del` and `only`. Note at most one rewrite step is performed. The method `auto` tries to apply many rewrite rules: it is a form of looping `simp`. In this sense, it can replace many `simp` steps, but it can also perform too many, leading to unexpected results. The method `blast` tries to find witnesses to existential statements by breadth-first search, and performs no simplification. The method `fact` allows us to state explicitly something which is implicitly carried around as context.

A more sophisticated mechanism than primitive recursion is available in *Isabelle*: inductive sets and predicates. It is possible to then do induction over an inductive set, or case analysis, by showing whichever property we like holds for canonical elements of the set. Inductive predicates check membership of an inductive set. As a basic example, consider the even numbers. We show the definition using an inductive set, and an inductive predicate:

inductive-set *even* :: *nat set*
where
zero: *0* ∈ *even*
| *sucsuc*: *n* ∈ *even* ⟹ (*n*+*2*) ∈ *even*

inductive *evenpred* :: *nat* ⇒ *bool*
where
zero: *evenpred 0*
| *sucsuc*: *evenpred n* ⟹ *evenpred* (*n*+*2*)

The sum of two even numbers is even. We show this by induction on the first element:

lemma *evenadd*:
assumes *n* ∈ *even*
and *m* ∈ *even*
shows (*n* + *m*) ∈ *even*
using *assms*
proof (*induct n*)
case *zero*
then show *0* + *m* ∈ *even* **using** ‹*m* ∈ *even*› **by** *auto*
next
case (*sucsuc n′*)
with ‹*m* ∈ *even*› **have** *n′* + *m* ∈ *even* **by** *simp*
then have (*n′*+*m*+*2*) ∈ *even* **using** *even.sucsuc*[**where** *n*=*n′*+*m*] **by** *auto*
then show (*n′* + *2*) + *m* ∈ *even* **by** *simp*
qed

3.4.1 Proof Theory in Isabelle

Isabelle is distributed with a package which allows one to encode sequent calculi. However, it does not allow one to reason about derivations, only check whether derivations are valid. To reason about derivations more ingenuity is needed. There has been little work done on formalisations of structural proof theory in *Isabelle*. A little over a decade ago, [Adams, 1997] noted that *Isabelle* was not sufficient for large scale meta-theory developments. We hope to show that this is no longer true.

Various results about signed interval logics were formalised in [Rasmussen, 2002]. An older dialogue of *Isabelle* was used, including ASCII syntax, and as a result the proofs are not especially readable. In fact, the statements of the theorems are readable[3], but the proof process itself is not.

Display logic [Belnap Jr., 1982] was embedded into *Isabelle* in [Dawson and Goré, 2001], and the same authors formalised a proof of *Cut* elimination in *Isabelle* in [Dawson and Goré, 2002]. Again, it is the fact that the results are formalised, rather than the readability of the formalisation, which is important. The work of [Dawson and Goré, 2002] also does not use the *inductive* predicates, instead formalising sequents and derivations using the **primrec** method. This has an unfortunate drawback: a function is needed to define when a derivation or sequent is *well-formed*. Using the *inductive* scheme, derivations and sequents are well-formed *by definition*. Indeed, it makes no sense discuss derivations which are not well-formed; they do not exist. Even if the proofs are not the object of study, it would still be possible to clean up [Dawson and Goré, 2002] using the *Isar* language, if only for the statement of the theorems (a normal apply script could be used to actually do the work).

A similar paper is that of [Ridge, 2006] (which will be discussed in some detail in section 4.3). The result formalised is Craig's Interpolation Theorem, which is a consequence of the admissibility of *Cut* for the calculus **G3c**. Again, the work uses **primrec** to define well-formed derivations and uses an apply script for the proof. Both decisions mean that the proof is longer than it needs to be and also not readable by a human (consisting of around 1000 lines of "apply" tactics). Indeed, it was the effort to adapt the proof of [Ridge, 2006] to intuitionistic first-order logic (**G3i**) which began the project described in this thesis.

3.5 Nominal Isabelle

One of the main benefits of using *Isabelle* is the package *Nominal Isabelle* [Tasson and Urban, 2005]. This is a formalisation of the work in [Pitts, 2003]. Naming and binding issues have been difficult to overcome in a formal system for some years. Most approaches relied on de Bruijn indices [Barendregt, 1981], [de Bruijn, 1972]. De Bruijn noted in [de Bruijn, 1972] that, while de Bruijn notation is "easy to handle in metalingual discussion" amd "easy for the computer and for the computer programmer" it is not claimed to be very good for the

[3] To one versed in the ASCII syntax of *Isabelle*.

human reader. Since we have the nice language *Isar* which produces something the casual mathematician can at least grasp, it is therefore imperative that we use a binding scheme which is likewise easy for humans to read.

Nominal Isabelle fulfils this criterion. Variables are as one would see them in a text book, as is binding. The *Nominal* method relies on swapping variables. Given a term, the set of variables which cannot be swapped arbitrarily for a new variable is called the *support* of that term. A variable is called *fresh* for a term if it is not in the support of that term. These notions are (at least partially) proved by the system after a datatype is created. There are some additional requirements placed upon the user: any primitive recursive definition on a nominal datatype requires proof obligations about freshness and support to be fulfilled. The freshness and support functions allow terms to be identified up to α-equivalence, which is a crucial feature of *Nominal Isabelle*, although it is not exploited here. As an example, consider the untyped lambda calculus [Barendregt, 1981]. We could define the terms by:

atom-decl *var*

nominal-datatype *lam* = *Var var*
| *App lam lam* (**infixr** · *65*)
| *Lam* ≪*var*≫*lam* (λ [-].-)

The notation **infixr** · 65 tells the system that the operator · is an infix operator, which associates to the right. To indicate an operator associates to the left, one uses **infixl**. The 65 is the precedence of the operator, in other words how tightly it binds. The lower the number, the higher the precedence.

We need to declare that there are special objects, called **atoms**, which can be bound. The ≪≫ notation tells the system that variables in that position will be bound, and we give an abbreviation []. Note how similar this looks to the treatment one would see in a text-book. The bound variable is not, strictly speaking, a variable, but rather a representative of an equivalence class. Consider the **S** combinator: the only difference in its appearance from a book would be the removal of the word *Var*, the removal of square brackets and (perhaps) the use of only one λ:

abbreviation
s-combinator (*S*)
where *S x y z* ≡ λ [*x*].(λ [*y*]. (λ [*z*]. (*Var x* · *Var z* · (*Var y* · *Var z*))))

3.5.1 Proof Theory in Nominal Isabelle

A few papers exist describing formalisations of proof theory using *Nominal Isabelle*. Given that it is a relatively recent development, that there are only a small number is to be expected.

In [Urban and Zhu, 2008] a strongly normalising cut-elimination procedure for classical logic was formalised. The proof uses a term notation and is, the authors note, "rather

complicated." The authors also note, however, that it would be unfeasible to prove the result in any other system using another approach. The benefits of formalisation are succinctly highlighted by [Urban and Zhu, 2008]: some cases in the informal proof (found in [Urban and Bierman, 2001] and [Urban, 2000]) were either missing or deficient. However, all such problems could be fixed and formalised.

In [Urban et al., 2007] an important result is described: a hole in the variable convention of Barendregt [Barendregt, 1981]. Whilst [Urban et al., 2007] does not contain the formalisation of this result, it was the formalisation which highlighted the gap in the first place. The details can be found at `http://isabelle.in.tum.de/nominal`.

As will be discussed in chapter 4, we have given a formalisation of Craig's Interpolation Theorem using *Nominal Isabelle* [Chapman et al., 2008] (4.3).

3.6 Results Formalised in Other Systems

The system Elf (which is superseded by Twelf [Pfenning and Schuermann, 2005]) is, like Coq, dependently typed. In [Pfenning, 2000], Cut elimination is proved in the absence of structural rules, for both classical and intuitionistic logic. The author notes that, in order to be fully appreciated, the reader needs "basic knowledge of...the Elf meta-language." Whilst this is true of any formalisation, to a certain extent, the meta-language of Elf is not easily readable by a human. A term structure is used for encoding derivations. However, the main drawback of the paper is that it did not verify that the "signatures implement meta-theoretic proofs"; in other words, whilst we have an algorithm which transforms cuts into smaller cuts, there is no guarantee that it is indeed correct. Pfenning notes it would be difficult to encode such a proof in a stronger framework, such as Coq, owing to the large number of additional lemmata one would have to prove in order to use de Bruijn indices and the explicit representation of contexts. He concludes that an "elegant representation of cut elimination in other systems is a non-trivial challenge which, we hope, others will take up" [Pfenning, 2000](§7).

In a similar vein, [Schürmann, 2000] uses the system Twelf to automate the meta-theory of deductive systems. It introduces a meta-logic $\mathcal{M}_2^+$ in which one can easily formalise meta-theory results using dependent types. It is implemented in Twelf; in effect, it gives Twelf the same capabilities for formalisation as *Isabelle* or Coq.

3.7 Conclusions

Given the state of theorem provers at the beginning of this project it seemed reasonable to use *Isabelle* for two reasons: the imperative style of *Isar* and the first-order package *Nominal Isabelle*. Both of these allow for proofs to be written in a style which is approaching that of a text book. Of course, there is some way to go before any mathematician can pick up an

Isar proof and understand all that goes on, but certainly there is more transparency than with a normal tactic script. A well-written *Isar* proof is surely a step forward in the effort to dovetail formal and informal proofs. It is this problem which will occupy the next chapter, and then the remainder of the thesis.

Chapter 4

Formalising Craig, Cut and Contraction

4.1 Introduction

Some formalisations of sequent calculi use terms. Whilst terms may encapsulate all of the information about a derivation, they are sometimes not as readable as one would wish. Term formalisations are very amenable, however, to proving results about the calculus; all the information about the derivation is at hand. It would be more satisfying to give a formalisation of a sequent calculus which is more akin to what one would see in a text book, such as [Troelstra and Schwichtenberg, 2000], but nevertheless is still useable for formalising results from structural proof theory. Thus, in this chapter we give some concrete examples of human-readable formalised calculi for which we prove non-trivial structural proof theory results. These examples are novel: *Contraction* admissibility for **G4ip** has never been formalised before, let alone in *Isar*. In the previous chapter, we saw examples of *Cut* admissibility formalised in other systems, but not in *Isar*. A formalisation of Craig's Interpolations Theorem was attempted in [Ridge, 2006] and [Boulmé, 1996], but neither was in *Isar*.

In section 4.2, we provide a method for specifying the formulae and rules of a calculus. Where it is important, it is easy for derivations to carry height information. In section 4.3, we formalise Craig's Interpolation Theorem [Takeuti, 1975], in an expansion of the work in [Chapman et al., 2008]. In section 4.4, we formalise *Cut* admissibility for **G3ip**, whilst in section 4.5, we formalise *Contraction* admissibility for **G4ip**.

4.2 Formalising Sequent Calculi

Sequent calculi are usually presented as rules of inference which can manipulate the formulae of a logic. One knows, under the assumption that the premisses of a rule of inference are roots of derivations, that the conclusion is likewise the root of a derivation. Furthermore, nothing else is the root of a derivation.[1] Such a characterisation of derivations is suited to *Isabelle*; roots of derivations are formed using *inductive predicates*.

The idea for sequent calculi is to give a syntax for sequents, and show how sequents are inductively formed. For instance, in a logic with conjunction, the rule:

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

may be plausible. In the language of inductive predicates, the symbol $\Rightarrow$ now carries more power; it yields the expression is well-formed. So, if $\Gamma \Rightarrow A$ and $\Gamma \Rightarrow B$, we really *know* that they are the roots of derivations. Thus, we know that from the pair of premisses, one can deduce $\Gamma \Rightarrow A \wedge B$ is the root of a derivation i.e. a sequent. In the formalisation, we use the symbol $\Rightarrow^\star$ to denote that a sequent is provable in this sense.

As an example, consider the rules for **G3ip** (see appendix A). Given a datatype for formulae it is a simple matter to encode the rules of the calculus:

datatype *form* = *Atom nat*
| *Imp form form* (-⊃-)
| *Conj form form* (- ∧* -)
| *Disj form form* (- ∨* -)
| *ff*

inductive
provable :: *form multiset* ⇒ *form* ⇒ *bool* (- ⇒* -)
where
Ax: ⟦(*Atom i*):# Γ⟧ ⟹ Γ ⇒* *Atom i*
| *LBot*: ⟦ *ff* :# Γ⟧ ⟹ Γ ⇒* *C*
| *ConjR*: ⟦Γ ⇒* *A* ; Γ ⇒* *B*⟧ ⟹ Γ ⇒* *A* ∧* *B*
| *ConjL*: ⟦Γ ⊕ *A* ⊕ *B* ⇒* *C*⟧ ⟹ Γ ⊕ *A*∧**B* ⇒* *C*
| *DisjR1*: ⟦Γ ⇒* *A*⟧ ⟹ Γ ⇒* *A*∨**B*
| *DisjR2*: ⟦Γ ⇒* *B*⟧ ⟹ Γ ⇒* *A*∨**B*
| *DisjL*: ⟦Γ ⊕ *A* ⇒* *C*; Γ ⊕ *B* ⇒* *C*⟧ ⟹ Γ ⊕ *A*∨**B* ⇒* *C*
| *ImpR*: ⟦ Γ ⊕ *A* ⇒* *B*⟧ ⟹ Γ ⇒* *A*⊃*B*
| *ImpL*: ⟦ Γ ⊕ *A*⊃*B* ⇒* *A* ; Γ ⊕ *B* ⇒* *C*⟧ ⟹ Γ ⊕ *A*⊃*B* ⇒* *C*

The syntax $A : \#\Gamma$ means A is a member of the multiset Γ. The notation $\Gamma \oplus A$ adds

[1] Some authors would have that any tree-like structure is a derivation, and further effort is required to decide what is a *valid* derivation. Similarly, some make the distinction between a formula and a well-formed formula: the former is just a string of symbols is a language, the latter obeys some particular structure. Here, we choose instead to have validity built-in.

the single formula A to the multiset of formulae Γ. Note that these sequents carry no information about their height or size. One cannot infer, automatically, the height or size of the derivation. We may augment each the previous definition with a natural number, here denoting the size of the derivation. Axioms and instances of $L\bot$ have size 0. Single premiss rules have size one greater than the size of the premiss, whereas two premiss rules have size one greater than the sum of the sizes of the premisses. The rules for **G4ip** (see appendix A) with size are shown:

inductive
provable-dp :: *form multiset* ⇒ *form* ⇒ *nat* ⇒ *bool* (- ⇒ - ↓ -)
where
Ax: $[\![(Atom\ i) :\# \Gamma]\!] \Longrightarrow \Gamma \Rightarrow Atom\ i \downarrow 0$
| *LBot*: $[\![ff :\# \Gamma]\!] \Longrightarrow \Gamma \Rightarrow C \downarrow 0$
| *ConjR*: $[\![\Gamma \Rightarrow A \downarrow n\ ;\ \Gamma \Rightarrow B \downarrow m]\!] \Longrightarrow \Gamma \Rightarrow A \wedge* B \downarrow n{+}m{+}1$
| *ConjL*: $[\![\Gamma \oplus A \oplus B \Rightarrow C \downarrow n]\!] \Longrightarrow \Gamma \oplus A \wedge* B \Rightarrow C \downarrow n{+}1$
| *DisjR1*: $[\![\Gamma \Rightarrow A \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \vee* B \downarrow n{+}1$
| *DisjR2*: $[\![\Gamma \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \vee* B \downarrow n{+}1$
| *DisjL*: $[\![\Gamma \oplus A \Rightarrow C \downarrow n\ ;\ \Gamma \oplus B \Rightarrow C \downarrow m]\!] \Longrightarrow \Gamma \oplus A \vee* B \Rightarrow C \downarrow n{+}m{+}1$
| *ImpR*: $[\![\ \Gamma \oplus A \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \supset B \downarrow n{+}1$

| *ImpL0*: $[\![\Gamma \oplus Atom\ i \oplus B \Rightarrow C \downarrow n]\!] \Longrightarrow \Gamma \oplus Atom\ i \oplus (Atom\ i \supset B) \Rightarrow C \downarrow n{+}1$
| *ImpLC*: $[\![\Gamma \oplus A \supset (B \supset C) \Rightarrow D \downarrow n]\!] \Longrightarrow \Gamma \oplus (A \wedge* B) \supset C \Rightarrow D \downarrow n{+}1$
| *ImpLD*: $[\![\Gamma \oplus A \supset C \oplus B \supset C \Rightarrow D \downarrow n]\!] \Longrightarrow \Gamma \oplus (A \vee* B) \supset C \Rightarrow D \downarrow n{+}1$
| *ImpLL*: $[\![\Gamma \oplus A \oplus B \supset C \Rightarrow B \downarrow n\ ;\ \Gamma \oplus C \Rightarrow D \downarrow m]\!] \Longrightarrow$
$\Gamma \oplus (A \supset B) \supset C \Rightarrow D \downarrow n{+}m{+}1$

The two calculi above use multisets for their contexts. It is possible to use other kinds of structures. Lists and sets are the most obvious candidates; in section 4.3, sets of formulae are used for contexts. No work has been done in this thesis using lists as contexts.

4.3 Formalising Craig's Interpolation Theorem

Maehara's proof [Takeuti, 1975] of the Craig Interpolation Theorem stands as one of the more beautiful and intricate consequences of *Cut* admissibility in the sequent calculus [Troelstra and Schwichtenberg, 2000] for first-order logic. Properly stated, it involves both the polarity of subformulae, *and* the first-order language of terms which may occur in the interpolant formula. The aim of this section is to present a formalised proof adapted for an intuitionistic **G3**-like system, in which routine informal considerations of free and bound variables in the language of first-order logic are rendered tractable by the use of *Nominal Isabelle*.

We build on the work of Ridge [Ridge, 2006] and Boulmé [Boulmé, 1996] and have formalised the result in *Nominal Isabelle* [Tasson and Urban, 2005]. The work of [Ridge, 2006] (for classical logic, formalised in *Isabelle*) is incomplete; there is a condition missing

in the statement of the theorem. This condition constrains the interpolant formula F for a sequent $\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$, where the two computed sequents are $\Gamma_1 \Rightarrow \Delta_1 \oplus F$ and $\Gamma_2 \oplus F \Rightarrow \Delta_2$, as follows: the language of F, denoted $\mathcal{L}(F)$, should be *common* to the languages $\mathcal{L}(\Gamma_1, \Delta_1)$ and $\mathcal{L}(\Gamma_2, \Delta_2)$. The definition of common language in [Ridge, 2006] accounts for the predicate constants (with their polarities), but *not* the free variables. This condition is the most difficult to formalise, but is likewise an important part of the correct statement of the theorem; the additional complexity in enforcing the condition arises inductively when one considers the rules for the quantifiers.

The work of Boulmé was undertaken in Coq [Coq Development Team, 2006], and used the locally-named syntax of McKinna and Pollack [McKinna and Pollack, 1993] to attempt to formalise Craig's theorem. This syntax was developed to deal with variable binding and reasoning up to α-conversion, as an alternative to de Bruijn notation. Substitution in a reduction rule was formalised by substituting a suitably fresh parameter (i.e. one that occured nowhere else in a particular derivation) for the bound variable, performing whichever reduction rule was needed, and then rebinding the original name. The method is constructive; a new fresh name need not actually be supplied, it is enough that such a fresh name actually exists, but the syntax stipulates that an explicit new parameter be given. Boulmé focused on a restricted language for classical first-order logic consisting of only *NAND* and $\forall$ (hence expressively complete). The interpolant for a particular case was shown to be valid for that case in individual lemmata, rather than as inductive cases of one theorem.

Here, we use instead the *Nominal Isabelle* system, with the intention of handling variable binding more cleanly, and show that the proof of the theorem for first-order intuitionistic logic can be formalised, including the tricky details that arise in the quantifier rule cases. The choice of intuitionistic logic simplifies the analysis of sequents $\Gamma \Rightarrow \Delta$, as Δ then consists of at most one formula. The nominal approach goes back to the work of Gabbay and Pitts [Gabbay and Pitts, 1999].

We intersperse the formal proofs with the proofs that one would normally see in a text book on proof theory. The informal proofs are displayed in a natural deduction style. We use the abbreviation Γ, Δ for $\Gamma \cup \Delta$ where a formula is displayed, and $\Gamma \oplus A$ where single formula A is to be added to a set Γ. That we use sets rather than multisets is notable. Firstly, the work in this section was an adaptation of the work of Ridge [Ridge, 2006] which used sets for contexts. Secondly, it shows that the choice of multisets, whilst useful, is not necessary: it is possible to formalise proof theory in *Isabelle* using sets for contexts.

4.3.1 The Development

The Formalisation

We build formulae as follows. An *atomic formula* is a predicate applied to a list of terms. We formalise a logic without equality; terms are simply variables, and thus zero-arity func-

tion symbols can be simulated by variables. The result would have been obscured beneath a mass of technical details were we to consider a logic with equality, or terms constructed of non-nullary function symbols. The interested reader is directed to [Troelstra and Schwichtenberg, 2000]. First-order formulae are built in the following way: propositional atoms are a predicate (with no arity) applied to a list of variables. Note that, since we are using intuitionistic logic, all of the logical connectives must be given as primitives. The propositional connectives are given in section 4.2. In the quantifier cases, the variable that is bound is more accurately called a representative of an α-equivalence class, although the fact that variables are identified up to α equivalence is never exploited here:

nominal-datatype *form* =
  Atom pred var list
| *ALL* ≪*var*≫*form* (∀∗ [-].-)
| *EX* ≪*var*≫*form* (∃∗ [-].-)

We have a notational shorthand for a quantification over a list of variables. These are "$\forall s$" and "$\exists s$" for the universal and existential quantifiers respectively. They are defined by primitive recursion on the list as follows:

consts
ALL-list :: *var list* ⇒ *form* ⇒ *form* (∀*s* [-].-)
primrec
∀*s* [*Nil*].*A* = *A*
∀*s* [*x*#*xs*].*A* = ∀∗ [*x*].(∀*s* [*xs*].*A*)

consts
EX-list :: *var list* ⇒ *form* ⇒ *form* (∃*s* [-].-)
primrec
∃*s* [*Nil*].*A* = *A*
∃*s* [*x*#*xs*].*A* = ∃∗ [*x*].(∃*s* [*xs*].*A*)

We use an aspect of the *Nominal Isabelle* package when we consider the individual constants of a formula. The function `freesf` returns the free variables of a formula, whereas `frees` returns the free variables of a set of formulae. One would ideally like to use the *support* function created when one defines a nominal datatype, however this was not possible here. The problem occurred when expanding the definition to the $\forall s$ and $\exists s$ quantifiers.

nominal-primrec *freesf* :: *form* ⇒ *var set*
  where
  freesf (*Atom n xs*) = *frees xs*
| *freesf* (*A* ∧∗ *B*) = (*freesf A*) ∪ (*freesf B*)
| *freesf* (*A* ∨∗ *B*) = (*freesf A*) ∪ (*freesf B*)
| *freesf* (*A* ⊃∗ *B*) = (*freesf A*) ∪ (*freesf B*)
| *freesf* (∀∗ [*x*].*A*) = (*freesf A*) − {*x*}
| *freesf* (∃∗ [*x*].*A*) = (*freesf A*) − {*x*}

```
| freesf (ff) = {}

lemma ALL-list-frees:
  shows freesf (∀s [L].E) = freesf E − (set L)
by (induct L) (auto)

lemma EX-list-frees:
  shows freesf (∃s [L]. E) = freesf E − (set L)
by (induct L) (auto)
```

The presence of implication as a primitive connective means that the positivity and negativity of a formula must be defined simultaneously. We use a pair of lists, the first list containing the positive predicates, and the second containing the negative predicates, as follows:

```
nominal-primrec pn :: form ⇒ (pred list × pred list)
where
  pn (Atom n xs) = ([n],[])
| pn (A ∧* B) = (let (pA,nA) = (pn A) in (let (pB,nB) = (pn B) in (pA@pB,nA@nB)))
| pn (A ∨* B) = (let (pA,nA) = (pn A) in (let (pB,nB) = (pn B) in (pA@pB,nA@nB)))
| pn (A ⊃* B) = (let (pA,nA) = (pn A) in (let (pB,nB) = (pn B) in (nA@pB,pA@nB)))
| pn (∀* [x].A) = pn A
| pn (∃* [x].A) = pn A
| pn (ff) = ([],[])
```

We also need to define capture avoiding substitution. This is straightforward when using *Nominal Isabelle*; the package allows us to say when a variable is fresh for another, denoted $x\sharp y$. For variables, this amounts to inequality: a variable is fresh for another if they are not equal. Here, we are really talking about α-equivalence classes, rather than individual variables. The notation $[t, x]A$ means we substitute the term t for the variable x in the formula A, whilst the notation $[t; x]ys$ forms the basis of this substitution: it substitutes a variable in a list of variables. Recall that our terms are simply variables:

```
nominal-primrec  subst-form :: var ⇒ var ⇒ form ⇒ form   where
  [z,y](Atom P xs) = Atom P ([z;y]xs)
| [z,y](A ∧* B)  = ([z,y]A) ∧* ([z,y]B)
| [z,y](A ∨* B)  = ([z,y]A) ∨* ([z,y]B)
| [z,y](A ⊃* B)    = ([z,y]A) ⊃* ([z,y]B)
| x♯(z,y) ⟹ [z,y](∀* [x].A) = ∀* [x].([z,y]A)
| x♯(z,y) ⟹ [z,y](∃* [x].A) = ∃* [x].([z,y]A)
| [z,y]ff = ff
```

We saw in section 4.2 how a provable sequent can be defined. Here, we use sets rather than multisets for context and have explicit *Weakening*. The propositional rules are as in

section 4.2 for **G3ip**, except that sets are used for contexts, which does not make a difference to the formulation of the rules. The first-order rules are:

inductive
provable :: *form set* ⇒ *form* ⇒ *bool* (- ⇒∗ -)
where
| *AllR*: ⟦x♯*frees* Γ; Γ ⇒∗ A⟧ ⟹ Γ ⇒∗ ∀∗ [x].A
| *ExR*: ⟦Γ ⇒∗ [y,x]A⟧ ⟹ Γ ⇒∗ ∃∗ [x].A
| *AllL*: ⟦(∀∗ [x].A)∈Γ; {[y,x]A}∪Γ ⇒∗ C⟧ ⟹ Γ ⇒∗ C
| *ExL*: ⟦(∃∗ [x].A)∈Γ; x♯(*frees* Γ ∪ *freesf* C); {A}∪Γ ⇒∗ C⟧ ⟹ Γ ⇒∗ C
| *wk*: ⟦Γ ⇒∗C⟧ ⟹ {A}∪Γ ⇒∗ C

As with the `freesf` function, one would prefer to say $x\sharp\Gamma$ in the rule *AllR*, however because we could not use the *support* function there, we cannot use the fresh function here.

Certain cases in the proof call for some derived rules. Since we have included *Weakening* as a primitive rule, we have generalised *Weakening* as a derived rule, so that if $\Gamma \Rightarrow^\star C$, and Γ is a subset of a finite set Γ', then $\Gamma' \Rightarrow^\star C$. More importantly, we have derived four rules which perform the appropriate quantifier rule over all the variables in a given list. As an example, here is the derived rule corresponding to $R\forall$:

$$\frac{\Gamma \Rightarrow^\star C}{\Gamma \Rightarrow^\star \forall s\ L.C}\ R\forall s$$

where freesf $L \cap$ frees $\Gamma = \emptyset$.

4.3.2 The Proof

We have formalised[2] the following theorem:

Theorem 5 (Craig's Interpolation Theorem) *Suppose that* $\Gamma \Rightarrow^\star C$. *Then, for any splitting of the context* $\Gamma \equiv \Gamma_1 \cup \Gamma_2$, *there exists an* E *such that:*

1. $\Gamma_1 \Rightarrow^\star E$ *and* $\Gamma_2, E \Rightarrow^\star C$.
2. *Any predicate that occurs positively in* E *occurs positively in* Γ_1 *and either positively in* C *or negatively in* Γ_2.
3. *Any predicate that occurs negatively in* E *occurs negatively in* Γ_1 *and either negatively in* C *or positively in* Γ_2.
4. $\mathit{frees}(E) \subseteq \mathit{frees}(\Gamma_1) \cap \mathit{frees}(\Gamma_2, C)$.

We use the notation $\Gamma_1; \Gamma_2 \overset{E}{\Longrightarrow} C$ to represent that E is a suitable *interpolant* for a splitting $\Gamma_1 \cup \Gamma_2$. We have formalised this as:

[2]The full formalised proof is available in the *Nominal Isabelle* distribution, at `http://isabelle.in.tum.de/nominal/`.

theorem *Craigs-Interpolation-Theorem*:
assumes a: $\Gamma_1 \cup \Gamma_2 \Rightarrow * \ C$
shows $\exists\, E.\ \Gamma_1 \Rightarrow * \ E \ \wedge \ \{E\} \cup \Gamma_2 \Rightarrow * \ C \ \wedge \ \Gamma_1, \Gamma_2, C \vdash E \ pnc$

where the notation "$\Gamma_1, \Gamma_2, C \vdash E$ pnc" is an abbreviation for E satisfying the conditions 2-4 with respect to Γ_1, Γ_2 and C. Normally, we would prove the theorem by induction on the height of the derivation of $\Gamma \Rightarrow C$, and then by case analysis on the last rule used in the derivation. This approach is possible in *Isabelle*, see [Ridge, 2006] for instance. Here we prove the theorem by induction on the "provable sequent" definition. This means we show the theorem is valid for the conclusion of each rule given that it is valid for the premisses. This induction scheme is derived and proved automatically by *Isabelle* when we write out inductive definitions.

Where a left rule was used to derive $\Gamma_1 \cup \Gamma_2 \Rightarrow^\star C$, there are two subcases: either the principal formula is in the left part of the split context, or it is in the right part. The splitting is not assumed to be disjoint, thus the two subcases are not mutually exclusive. In other words, it is in Γ_1 or Γ_2. When we refer to "the left case" and "the right case", we really mean that "the principal formula is in the left part of the split context..." etc. Since we have a single succedent calculus, we have no such splitting when using right rules. This leads to a total of 22 subcases: 4 base cases, 10 cases from left rules, 5 cases from right rules, and 3 *Weakening* cases.

In what follows, the names of the subsections refer to the rule(s) used in deriving the provable sequent. The use of variable binding is only evident in the first-order cases, therefore we give only sketches of the propositional cases, where there is no binding. They are still fully formalised, but the output is suppressed.

Axioms and $L\bot$

In the case where the derivation of $\Gamma_1, \Gamma_2 \Rightarrow^\star C$ is an axiom, there are two cases. The left has $C \in \Gamma_1$ and the right has $C \in \Gamma_2$. In the former, we need to find a formula E such that:

$$\Gamma_1 \Rightarrow^\star E \text{ and } E, \Gamma_2 \Rightarrow^\star C$$

A suitable candidate is $E \equiv C$, which would make both provable sequents instances of Ax. We must also check that $\Gamma, \Gamma', C \vdash C$ pnc, which is trivially true. For this case we can conclude $\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} C$.

In the other case, we require an E so that:

$$\Gamma_1 \Rightarrow^\star E \text{ and } E, \Gamma_2 \Rightarrow^\star C$$

which only hold for general Γ_1 if we have $E \equiv \bot \supset \bot$; in other words, $\top$. Since $\bot$ has neither free variables nor predicate symbols, the condition $\Gamma_1, \Gamma_2, C \vdash \bot \supset \bot$ pnc is trivially true for

any Γ_1 and Γ_2. Therefore, for this case we have $\Gamma_1;\Gamma_2 \overset{\bot\supset\bot}{\Longrightarrow} C$.

Likewise, the provable sequents we require in the case where the rule used is $L\bot$ are straightforward. We have two subcases, where $\bot$ is either in Γ_1 or Γ_2. In the former, we have that the interpolant is $\bot$, and in the latter we have the interpolant is $\bot\supset\bot$:

case (*LBot* Γ C)
then have *a1*: *finite* $\Gamma_1 \wedge$ *finite* Γ_2
  and *a2*: $ff \in \Gamma_1 \cup \Gamma_2$ **by** *simp-all*
have $ff \in \Gamma_1 \vee ff \in \Gamma_2$ **using** *a2* **by** *blast*
moreover
{assume $ff \in \Gamma_1$
 with *a1* **have** $\exists E.\ \Gamma_1 \Rightarrow* E \wedge \{E\}\cup\Gamma_2 \Rightarrow* C \wedge \Gamma_1,\Gamma_2,C \vdash E$ *pnc* **by** *auto*
}
moreover
{assume *b2*: $ff \in \Gamma_2$
 with *a1* **have** $\Gamma_1 \Rightarrow* ff \supset* ff$
    and $\{ff \supset* ff\}\cup\Gamma_2 \Rightarrow* C$
    and $\Gamma_1,\Gamma_2,C \vdash (ff \supset* ff)$ *pnc*
  by (*auto*)
 then have $\exists E.\ \Gamma_1 \Rightarrow* E \wedge \{E\}\cup\Gamma_2 \Rightarrow* C \wedge \Gamma_1,\Gamma_2,C \vdash E$ *pnc* **by** *blast*
}
ultimately show $\exists E.\ \Gamma_1 \Rightarrow* E \wedge \{E\}\cup\Gamma_2 \Rightarrow* C \wedge \Gamma_1,\Gamma_2,C \vdash E$ *pnc* **by** *blast*

$R\supset$ and $L\supset$

For the right rule, however we split the antecedent the same argument applies, since there is no distinguished formula in it. We split the premiss on the right, and suppose the interpolant of the premiss is E. We therefore have the two sequents $\Gamma_1 \Rightarrow^\star E$ and $\Gamma_2 \oplus A \oplus E \Rightarrow^\star B$. Leaving the first alone, we have the simple deduction for the second:

$$\frac{\Gamma_2, A, E \Rightarrow^\star B}{\Gamma_2, E \Rightarrow^\star A \supset B}\ R\supset$$

which gives us the interpolant for the whole as E:

$$\frac{\Gamma_1; A, \Gamma_2 \overset{E}{\Longrightarrow} B}{\Gamma_1;\Gamma_2 \overset{E}{\Longrightarrow} A \supset B}$$

There are two subcases for $L\supset$. The left subcase has $A \supset B \in \Gamma_1$ and is the most unusual of the propositional cases. Since the statement of the theorem says "for *any* splitting of the context", this means from our induction hypothesis we can choose whichever splitting we want. In this case, we choose a different splitting for the premisses than for the conclusion. Some brief experimentation reveals that we should split the first premiss as Γ_2 and Γ_1 and

the second as $\Gamma_1 \cup B$ and Γ_2. In the formalisation below, we have instantiated the induction hypotheses to reflect this. This gives us four sequents: $\Gamma_2 \Rightarrow^\star E_1$ and $\Gamma_1 \oplus E_1 \Rightarrow^\star A$ and $\Gamma_1 \oplus B \Rightarrow^\star E_2$ and $\Gamma_2 \oplus E_2 \Rightarrow^\star C$.

Taking the first and fourth of these we can create the deduction:

$$\dfrac{\dfrac{\Gamma_2 \Rightarrow^\star E_1}{\Gamma_2, E_1 \supset E_2 \Rightarrow^\star E_1}\, w \qquad \Gamma_2, E_2 \Rightarrow^\star C}{\Gamma_2, E_1 \supset E_2 \Rightarrow^\star C}\, L\supset$$

whereas using the second and third we can create the deduction:

$$\dfrac{\Gamma_1, E_1 \Rightarrow^\star A \quad \dfrac{\dfrac{\Gamma_1, B \Rightarrow^\star E_2}{B, \Gamma_1, E_1 \Rightarrow^\star E_2}\, w}{\Gamma_1, E_1 \Rightarrow^\star E_2}\, L\supset}{\Gamma_1 \Rightarrow^\star E_1 \supset E_2}\, R\supset$$

This is precisely the form we need, with the interpolant being $E_1 \supset E_2$. We can therefore conclude that the following is a valid deduction for this case, recalling that $A \supset B \in \Gamma_1$:

$$\dfrac{\Gamma_2; \Gamma_1 \overset{E_1}{\Longrightarrow} A \quad \Gamma_1, B; \Gamma_2 \overset{E_2}{\Longrightarrow} C}{\Gamma_1; \Gamma_2 \overset{E_1 \supset E_2}{\Longrightarrow} C}$$

We can see this formalised in the following fragment, where the language conditions are also verified:

```
case (ImpL A B Γ C Γ₁ Γ₂)
  assume b1: (A⊃*B)∈Γ₁
  have ihL: ∃ E. Γ₂ ⇒* E ∧ ({E} ∪ Γ₁) ⇒* A ∧
                Γ₂,Γ₁,A ⊢ E pnc  by (simp)
  have ihR: ∃ E. ({B}∪Γ₁) ⇒* E ∧ ({E} ∪ Γ₂) ⇒* C ∧
                ({B}∪Γ₁),Γ₂,C ⊢ E pnc  by (simp)
  from ihL ihR obtain E1 E2
    where c1: Γ₂ ⇒* E1 and c2: {E1}∪Γ₁ ⇒* A
      and d1: {B}∪Γ₁ ⇒* E2 and d2: {E2}∪Γ₂ ⇒* C
      and c3: Γ₂,Γ₁,A ⊢ E1 pnc and d3: ({B}∪Γ₁),Γ₂,C ⊢ E2 pnc by auto
  from d1 have {B,E1}∪Γ₁ ⇒* E2 using provable.wk by (blast)
  then have {E1}∪Γ₁ ⇒* E2 using b1 c2 provable.ImpL by (auto)
  then have Γ₁ ⇒* E1 ⊃* E2 using provable.ImpR by auto
  moreover
  from c1 d2 have {E1 ⊃* E2}∪ Γ₂ ⇒* E1
            and {E1 ⊃* E2,E2}∪Γ₂ ⇒* C by (blast)+
  then have {E1⊃*E2}∪Γ₂ ⇒* C using provable.ImpL by (auto)
  moreover
  from c3 d3 have Γ₁,Γ₂,C ⊢ E1 ⊃* E2 pnc using b1  by (auto)
```

ultimately have $\exists E.\ \Gamma_1 \Rightarrow\!* \ E \wedge \{E\} \cup \Gamma_2 \Rightarrow\!* \ C \wedge \Gamma_1, \Gamma_2, C \vdash E$ *pnc* **by** *blast*

In the right case ($A \supset B \in \Gamma_2$), assuming via the induction hypothesis that the first premiss has interpolant E_1 and the second premiss interpolant E_2, we split both premisses on the right, so $\Gamma_1; \Gamma_2 \overset{E_1}{\Longrightarrow} A$ and $\Gamma_1; \Gamma_2 \oplus B \overset{E_2}{\Longrightarrow} C$. We then obtain four sequents: $\Gamma_1 \Rightarrow^\star E_1$ and $\Gamma_2 \oplus E_1 \Rightarrow^\star A$ and $\Gamma_1 \Rightarrow^\star E_2$ and $\Gamma_2 \oplus B \oplus E_2 \Rightarrow^\star C$. Naturally, we pair them up according to contexts. The first and third premisses therefore give:

$$\frac{\Gamma_1 \Rightarrow^\star E_1 \quad \Gamma_1 \Rightarrow^\star E_2}{\Gamma_1 \Rightarrow^\star E_1 \wedge E_2}\ R\wedge$$

whereas the remaining two sequents give:

$$\frac{\dfrac{\dfrac{\Gamma_2, E_1 \Rightarrow^\star A}{\Gamma_2, E_1, E_2 \Rightarrow^\star A}\ w \quad \dfrac{\Gamma_2, B, E_2 \Rightarrow^\star C}{\Gamma_2, B, E_1, E_2 \Rightarrow^\star C}\ w}{\Gamma_2, E_1, E_2 \Rightarrow^\star C}\ L\supset}{\Gamma_2, E_1 \wedge E_2 \Rightarrow^\star C}\ L\wedge$$

which gives us the required interpolant as $E_1 \wedge E_2$:

$$\frac{\Gamma_1; \Gamma_2 \overset{E_1}{\Longrightarrow} A \quad \Gamma_1; B, \Gamma_2 \overset{E_2}{\Longrightarrow} C}{\Gamma_1; \Gamma_2 \overset{E_1 \wedge E_2}{\Longrightarrow} C}$$

Note that a conjunctive formula is the interpolant for rules involving implication. Thus, this would not be a valid interpolant if we were to restrict the language to the implicational fragment. Craig's Interpolation theorem for the implicational fragment of intuitionistic logic is given in [Kanazawa, 2006], but is given in semantic form, rather than proof theoretic form, and so would not fit into the proof described here.

$R\wedge$ and $L\wedge$

For $R\wedge$, whichever way we split the antecedent the same argument applies, and likewise for the premisses. Therefore, assuming that the interpolant for the first premiss is E_1 and the interpolant for the second premiss is E_2, we get four sequents: $\Gamma_1 \Rightarrow^\star E_1$ and $\Gamma_2 \oplus E_1 \Rightarrow^\star A$ and $\Gamma_1 \Rightarrow^\star E_2$ and $\Gamma_2 \oplus E_2 \Rightarrow^\star B$. Pairing them up by context, we get:

$$\frac{\Gamma_1 \Rightarrow^\star E_1 \quad \Gamma_1 \Rightarrow^\star E_2}{\Gamma_1 \Rightarrow^\star E_1 \wedge E_2}\ R\wedge$$

and:

$$\frac{\dfrac{\dfrac{\Gamma_2, E_1 \Rightarrow^\star A}{\Gamma_2, E_1, E_2 \Rightarrow^\star A}\ w \quad \dfrac{\Gamma_2, E_2 \Rightarrow^\star B}{\Gamma_2, E_1, E_2 \Rightarrow^\star B}\ w}{\Gamma_2, E_1, E_2 \Rightarrow^\star A \wedge B}\ R\wedge}{\Gamma_2, E_1 \wedge E_2 \Rightarrow^\star A \wedge B}\ L\wedge$$

which means that $E_1 \wedge E_2$ is the interpolant:

$$\frac{\Gamma_1;\Gamma_2 \overset{E_1}{\Longrightarrow} A \quad \Gamma_1;\Gamma_2 \overset{E_2}{\Longrightarrow} B}{\Gamma_1;\Gamma_2 \overset{E_1 \wedge E_2}{\Longrightarrow} A \wedge B}$$

The two subcases for $L\wedge$ are simple. For the left case, assume that the interpolant is E, and split A, B likewise on the left, we get the two sequents $\Gamma_1 \oplus A \oplus B \Rightarrow^\star E$ and $\Gamma_2 \oplus E \Rightarrow^\star C$. We leave the second of these alone, and taking the first apply $L\wedge$. We can then conclude that E is the interpolant:

$$\frac{\Gamma_1, A, B;\Gamma_2 \overset{E}{\Longrightarrow} C}{\Gamma_1;\Gamma_2 \overset{E}{\Longrightarrow} C}$$

The right case is symmetrical, therefore E, the interpolant supplied by the induction hypothesis, is also the interpolant for the conclusion.

$R\vee$ and $L\vee$

We have two rules for $R\vee$. However, the two cases are almost identical, so we will only show one. Whichever way we split the antecedent of the conclusion, the same reasoning applies, and likewise for the premiss. Suppose the interpolant from the induction hypothesis is E, and assume further that we used the rule $R\vee_1$. Then we have the sequents $\Gamma_1 \Rightarrow^\star E$ and $\Gamma_2 \oplus E \Rightarrow^\star A$. Using the rule $R\vee_1$ on the second, we obtain $\Gamma_2 \oplus E \Rightarrow^\star A \vee B$. Therefore the interpolant in this case is E, and is given by the deduction:

$$\frac{\Gamma_1;\Gamma_2 \overset{E}{\Longrightarrow} A}{\Gamma_1;\Gamma_2 \overset{E}{\Longrightarrow} A \vee B}$$

We get the same result if $R\vee_2$ was used in both situations.

Now we consider $L\vee$. In the left case assume that the interpolant for the first premiss is E_1, and the interpolant for the second premiss is E_2. Now, split both of the premisses on the left, to obtain the two sequents from the left premiss $\Gamma_1 \oplus A \Rightarrow^\star E_1$ and $\Gamma_2 \oplus E_1 \Rightarrow^\star C$, and the two sequents from the right premiss $\Gamma_1 \oplus B \Rightarrow^\star E_2$ and $\Gamma_2 \oplus E_2 \Rightarrow^\star C$. Again, pairing up by contexts, we have:

$$\frac{\Gamma_2, E_1 \Rightarrow^\star C \quad \Gamma_2, E_2 \Rightarrow^\star C}{\Gamma_2, E_1 \vee E_2 \Rightarrow^\star C}\ L\vee$$

and:

$$\dfrac{\dfrac{\Gamma_1, A \Rightarrow^\star E_1}{\Gamma_1, A \Rightarrow^\star E_1 \vee E_2}\, R\vee \quad \dfrac{\Gamma_1, B \Rightarrow^\star E_2}{\Gamma_1, B \Rightarrow^\star E_1 \vee E_2}\, R\vee}{\Gamma_1 \Rightarrow^\star E_1 \vee E_2}\, L\vee$$

This means the required interpolant is $E_1 \vee E_2$.

For the right case, we again split both premisses on the right, so the following deductions suffice, assuming that E_1 and E_2 are the interpolants:

$$\dfrac{\Gamma_1 \Rightarrow^\star E_1 \quad \Gamma_1 \Rightarrow^\star E_2}{\Gamma_1 \Rightarrow^\star E_1 \wedge E_2}\, R\wedge$$

and:

$$\dfrac{\dfrac{\dfrac{\Gamma_2, A, E_1 \Rightarrow^\star C}{\Gamma_2, A, E_1, E_2 \Rightarrow^\star C}\, w \quad \dfrac{\Gamma_2, B, E_2 \Rightarrow^\star C}{\Gamma_2, B, E_1, E_2 \Rightarrow^\star C}\, w}{\Gamma_2, E_1, E_2 \Rightarrow^\star C}\, L\vee}{\Gamma_2, E_1 \wedge E_2 \Rightarrow^\star E}\, L\wedge$$

meaning that $E_1 \wedge E_2$ is the interpolant.

R$\exists$

A first attempt at finding an interpolant for this case would use the interpolant supplied by the induction hypothesis. Whilst it would give us the two provable sequents that we need for the theorem, this interpolant fails the language condition for the conclusion. Suppose the induction hypothesis gives us the two provable sequents $\Gamma_1 \Rightarrow^\star E$ and $\Gamma_2 \oplus E \Rightarrow^\star [y, x]A$. The induction hypothesis gives us that the free variables of E are contained in the free variables of Γ_1 and the free variables of $\Gamma_2 \oplus [y, x]A$. Suppose that there were some free variables in E that were in the free variables of y, but *not* in the free variables of Γ_2 or A. These free variables would no longer appear in the conclusion, and so the language condition would fail when using E. This is the crucial difference between our definitions and formalisation and those in [Ridge, 2006]: E would be a valid interpolant in that formalisation. We need to remove these free variables, which we do by quantification. In this case, we use existential quantification.

Let the set of such variables be L. Since they are finite, indeed there is at most one such variable, we can form a list from this set, which we will also call L, in a slight abuse of notation. Since $R\exists$ has no side conditions about the freshness of variables, we can apply $R\exists$ for every variable in the list:

$$\dfrac{\Gamma_1 \Rightarrow^\star E}{\Gamma_1 \Rightarrow^\star \exists s\ L.E}\, R\exists s$$

On the second sequent, we can apply the derived rule $L\exists s$, *after* applying $R\exists$:

$$\dfrac{\dfrac{\Gamma_2, E \Rightarrow^\star [y,x]A}{\Gamma_2, E \Rightarrow^\star \exists x A}\, R\exists}{\Gamma_2, \exists s\ L.E \Rightarrow^\star \exists x A}\, L\exists s$$

We can see this argument formalised as follows:

case (*ExR* Γ y x A)
then have *a1*: $\Gamma_1 \cup \Gamma_2 \Rightarrow* [y,x]A$
  and *ih*:$\exists\, E.\ \Gamma_1 \Rightarrow* E \wedge \{E\} \cup \Gamma_2 \Rightarrow* [y,x]A \wedge \Gamma_1,\Gamma_2,[y,x]A \vdash E$ *pnc* **by** *simp-all*
from *ih* **obtain** E **where** *b1*: $\Gamma_1 \Rightarrow* E$
  and *b2*: $\{E\} \cup \Gamma_2 \Rightarrow* [y,x]A$
  and *b3*: $\Gamma_1,\Gamma_2,[y,x]A \vdash E$ *pnc* **by** *blast*
have *finite* ((*freesf* E) $-$ ((*frees* Γ_2) $\cup$ *freesf* $(\exists* [x].A)$)) **by** (*simp*)
then obtain L **where** *eq*: *set* L = (*freesf* E) $-$ ((*frees* Γ_2) $\cup$ (*freesf* $(\exists* [x].A)$))
  using *exists-list-for-finite-set* **by** *auto*
from *b1* **have** $\Gamma_1 \Rightarrow* \exists s\ [L].E$ **by** (*rule exists-right-intros*)
moreover
from *b2* **have** $\{E\} \cup \Gamma_2 \Rightarrow* \exists* [x].A$ **using** *provable.ExR* **by** *auto*
then have $\{\exists s\ [L].E\} \cup \Gamma_2 \Rightarrow* \exists* [x].A$ **using** *eq* **by** (*rule-tac exists-left-intros*)

moreover
from *b3* **have** $\Gamma_1,\Gamma_2,\exists* [x].A \vdash \exists s\ [L].E$ *pnc* **using** *eq* **by** (*auto*)
ultimately show $\exists\, E.\ \Gamma_1 \Rightarrow* E \wedge \{E\} \cup \Gamma_2 \Rightarrow* \exists* [x].A \wedge$
  $\Gamma_1,\Gamma_2,\exists* [x].A \vdash E$ *pnc* **by** *blast*

$L\forall$

We have the same problem in this case as in the case for $R\exists$. In the left subcase, we define L as the list of variables which appear free in y and not free in $\forall x A$ and $Gamma_1$. There are two provable sequents supplied by the induction hypothesis, namely $\Gamma_1 \oplus [y,x]A \Rightarrow^\star E$ and $\Gamma_2 \oplus E \Rightarrow^\star C$, with $\forall x A \in \Gamma_1$. Using the former, we first apply $L\forall$, and then, since we know that the variables in L appear nowhere free in Γ_1 by construction, then we can apply one instance of $R\forall$ for every variable in L:

$$\dfrac{\dfrac{\Gamma_1, [y,x]A \Rightarrow^\star E}{\Gamma_1 \Rightarrow^\star E}\, L\forall}{\Gamma_1 \Rightarrow^\star \forall s\ L.E}\, R\forall s$$

Using the other sequent, we can simply apply our derived rule $L\forall s$, thus $\forall s\ L.E$ is the required interpolant.

In the right case, we define L as the list of variables not free in $\Gamma_2, \forall x A$ and C. The interesting sequent is now the second obtained from the induction hypothesis, $\Gamma_2 \oplus [y,x]A \oplus$

$E \Rightarrow^\star C$. We first apply $L\forall$, and then we know that the variables in L do not appear free in the new sequent, therefore we can apply $L\exists$ for each of the variables in L:

$$\dfrac{\dfrac{\Gamma_2, [y, x]A, E \Rightarrow^\star C}{\Gamma_2, E \Rightarrow^\star C}\, L\forall}{\Gamma_2, \exists s\ L.E \Rightarrow^\star C}\, L\exists s$$

The proof for the left case is formalised as follows:

case (*AllL x A* Γ *y C* Γ_1 Γ_2) **assume** *b1*: $\forall * [x].A \in \Gamma_1$
have *ih*: $\exists\ E.\ \{[y,x]A\} \cup \Gamma_1 \Rightarrow * E \wedge \{E\} \cup \Gamma_2 \Rightarrow * C \wedge$
$\{[y,x]A\} \cup \Gamma_1, \Gamma_2, C \vdash E$ *pnc* **by** *auto*
from *ih* **obtain** *E* **where**
c1: $\{[y,x]A\} \cup \Gamma_1 \Rightarrow * E$
and *c2*: $\{E\} \cup \Gamma_2 \Rightarrow * C$
and *c3*: $\{[y,x]A\} \cup \Gamma_1, \Gamma_2, C \vdash E$ *pnc* **by** *auto*
have *finite* (*freesf E − frees* Γ_1) **by** (*simp*)
then obtain *L* **where**
eq: *set L* = *freesf E − frees* Γ_1 **using** *exists-list-for-finite-set* **by** *auto*
then have *set L* ∩ *frees* $\Gamma_1 = \{\}$ **by** *auto*
from *c1* **have** $\Gamma_1 \Rightarrow * E$ **using** *provable.AllL* ‹$\forall * [x].A \in \Gamma_1$› **by** *auto*
then have $\Gamma_1 \Rightarrow * \forall s\ [L].E$ **using** ‹*set L* ∩ *frees* $\Gamma_1 = \{\}$› **by** (*rule forall-right-intros*)
moreover
from *c2* **have** $\{\forall s\ [L].E\} \cup \Gamma_2 \Rightarrow * C$ **by** (*rule forall-left-intros*)
moreover
from *c3* **have** $\Gamma_1, \Gamma_2, C \vdash \forall s\ [L].E$ *pnc* **using** *eq b1*
by (*auto*)
ultimately have $\exists\ E.\ \Gamma_1 \Rightarrow * E \wedge \{E\} \cup \Gamma_2 \Rightarrow * C \wedge \Gamma_1, \Gamma_2, C \vdash E$ *pnc* **by** *blast*

$R\forall$ and $L\exists$

In the case of $R\forall$, it does not matter how we split the conclusion and premiss. Thus, we have the provable sequents, supplied by the induction hypothesis, $\Gamma_1 \Rightarrow^\star E$ and $\Gamma_2 \oplus E \Rightarrow^\star A$, and further that $x \notin \text{frees}(\Gamma_1, \Gamma_2, E)$. This means that we can simply apply $R\forall$ to the second of these two provable sequents, and then have the required sequents. Furthermore, since we know that $x \notin \text{frees}(E)$, we have that the free variables of E are contained in the free variables of $\Gamma_2 \oplus \forall x A$.

The two cases for $L\exists$ are symmetrical to that of $R\forall$; the induction hypothesis supplies that the quantified variable will be not be in the free variables of the interpolant, and so we can just apply the rule $L\exists$ to the appropriate sequent, leaving the other alone.

The *Weakening* cases are uninteresting and so are not shown. All of the cases have now been shown, and the proof of the theorem is complete.

4.3.3 Mechanisation Statistics and Conclusions

The current work stands at 823 lines [3], including white space and comments. Except for the *Weakening* case, all of the cases were informally sketched. Roughly a quarter of the formal proof was shown. The number of proof steps is 779. The proof in [Ridge, 2006] for classical logic was adapted for intuitionistic logic, and that comprised 1002 lines; moreover the theorem in that work was not as powerful as the one in the current work, owing to the absence of the condition on free variables. We also use the more verbose *Isar* language, and not a tactic script, which necessarily adds to the length of our proof. The main insight that made this proof much shorter was the removal of explicitly mentioning derivations, in favour of the notion of a provable sequent. As an example, a proof of the theorem in this section was attempted using explicit derivations, and the work was around 1200 lines. The use of *Nominal Isabelle* also allowed us to reason clearly and simply about capture-avoiding substitution.

We used *Weakening* (*wk* from the definition in section 6.8) as a primitive rule. However, we could have removed this primitive rule and instead shown *Weakening* was admissible. The purpose of this formalisation was to prove Craig's Interpolation Theorem, not the admissibility of structural rules for first-order logics. Thus, *wk* was kept as a primitive rule.

It would be a relatively straightforward to adapt this development and proof for classical logic without equality. We would need more subcases for each rule, since a sequent calculus for classical logic permits sets, or multisets, of formulae for succedents (see for instance [Troelstra and Schwichtenberg, 2000]). However, we could also interdefine the connectives, meaning one needs to consider fewer rules. It should be possible to extend the result to a logic with equality and non-nullary function symbols, although we are unaware that such a formalisation has been performed.

Interpolation results form an important part of computer science. They can be applied to type-checking in C programs, as shown in [Jhala et al., 2007], and also to model checking, as in [McMillan, 2005], amongst other things.

4.4 Formalising Cut Admissibility

4.4.1 An Induction Measure

For the proof of Craig's Interpolation Theorem, we performed induction on the inductive predicate `provable`. The inductive predicate hides information about the height of the derivation. For (this proof of) *Cut* admissibility for propositional logic using multisets, however, we need height to be explicit; the induction measure used is a lexicographic order of the complexity or length of the cut formula followed by the height of the instance of

[3] available as part of the *Nominal Isabelle* distribution at "http://isabelle.in.tum.de/nominal/"

Cut. We saw in section 4.2 how to encode an inductive definition of a provable sequent with height. In [Troelstra and Schwichtenberg, 2000] the *length* of a formula A, denoted $s(A)$ was defined as:

$$\begin{array}{lcll} s(\bot) & = & 0 & \\ s(P) & = & 1 & \text{for atomic } P \\ s(\circ A) & = & s(A)+1 & \text{for unary } \circ \\ s(A \circ B) & = & s(A)+s(B)+1 & \text{for binary } \circ \end{array}$$

We instead use a different definition for compound formulae, that of *depth*:

$$s(\circ[A_1,\ldots,A_n]) = \max_n s(A_i) + 1$$

This is easily formalised within *Isabelle*. Because **G3ip** only uses binary connectives this can be done with an `if-then-else` clause. Note that we have made no distinction between propositional atoms and $\bot$:

```
consts length :: form ⇒ nat
primrec
  length (At i) = 0
  length (A ⊃ B) =
      (if (length A ≤ length B) then (length B + 1) else (length A + 1))
  length (A ∧* B) =
      (if (length A ≤ length B) then (length B + 1) else (length A + 1))
  length (A ∨* B) =
      (if (length A ≤ length B) then (length B + 1) else (length A + 1))
  length (ff) = 0
```

The lexicographic order is a measure on a pair of natural numbers, then. *Isabelle* allows one to define induction principles from scratch. All one must do to use a user-defined measure is show that it is well-founded. That natural numbers are well-founded is part of the *Isabelle* distribution, so the burden on the user is somewhat reduced. The following is the proof of well-foundedness for a pair of natural numbers:

```
abbreviation
less-prod-nat (- <* -) where
   p <* q ≡ (p,q) : less-than <*lex*> less-than

lemma nat-prod-induct [case-names less]:
 fixes x y :: nat
 assumes induct-step: ⋀ x y. (⋀ u v. (u, v) <* (x, y) ⟹ P u v) ⟹ P x y
 shows P x y
proof −
 have wf (less-than <*lex*> less-than) by blast
 then show ?thesis
```

```
proof (induct p ≡ (x, y) arbitrary: x y)
  case (less p)
  show P x y
  proof (rule induct-step)
    fix u v
    assume (u, v) <* (x, y)
    with less show P u v by simp
  qed
 qed
qed
```

Much like strong induction for the natural numbers, there is no base case. Thus, most proofs using this induction measure will contain a sub-proof where case-analysis is performed.

4.4.2 A Proof of Cut Admissibility for G3ip

It is common for text-books, such as [Troelstra and Schwichtenberg, 2000] and [Negri and von Plato, 2001], to prove *Cut* admissibility by inspecting whether or not the cut formula is principal for the left or right premiss of the instance of *Cut*. To closely follow such a proof would require us to give a full definition of what it means to be principal for a derivation. Rather than that, we prove the result by case analysis on the last rule used in the left premiss of the instance of *Cut*. As we shall see, this approach relies heavily on the invertibility of the rules of **G3ip**.

Another result which we had to prove is the admissibility (depth-preserving) of *Weakening*. An example of the inversion result, along with depth-preserving *Weakening* and its natural extension, are shown without proof:

```
lemma inversionDisjL:
  assumes Γ ⊕ A∨*B ⇒ C ↓ n
  shows ∃ j k. j≤n ∧ k≤n ∧ Γ ⊕ A ⇒ C ↓ j ∧ Γ ⊕ B ⇒ C ↓ k

lemma dpWeak:
  fixes Γ :: form multiset
  assumes Γ ⇒ C ↓ n
shows Γ ⊕ A ⇒ C ↓ n

lemma dpWeak':
  assumes Γ ⇒ C ↓ n
  shows Γ + Γ' ⇒ C ↓ n
```

As with the previous section, we will intersperse the sketches of the cases with formal proof blocks.

lemma *cutAdmissibility*:
fixes A :: *form*
  and n m :: *nat*
assumes $\Gamma \Rightarrow A \downarrow n$ **and** $\Gamma \oplus A \Rightarrow C \downarrow m$
shows $\exists\ k.\ \Gamma \Rightarrow C \downarrow k$
using *assms*
proof (*induct x* ≡ *length A y* ≡ *n+m+1 rule*: *nat-prod-induct*)

The proof method is induction, introduced by the keyword *induct*. We use the rule *nat-product-induct*, shown on the previous page. We have to instantiate the variables in that method, which we do by the notation $x \equiv$ length A and $y \equiv n + m + 1$. In other words we set the lexicographic induction up using the length of the *Cut* formula, and then the sum of the heights of the premisses.

There is only one case to consider. So, using the induction hypothesis, we prove the case by a split on the last rule used for the left premiss:

case (*less x y*)
then have *IH*: $\bigwedge\ u\ v\ \Gamma\ A\ n\ C\ m.$
  $[\![(u,\ v) <*\ (x,\ y);\ \Gamma \Rightarrow A \downarrow n;\ \Gamma \oplus A \Rightarrow C \downarrow m;\ u = \mathit{length}\ A;\ v = n + m + 1]\!]$
  $\Longrightarrow \exists\, a.\ \Gamma \Rightarrow C \downarrow a$ **by** *auto*
have $\Gamma \Rightarrow A \downarrow n$ **by** *fact*
then show *?case*
  proof (*cases*)

This gives 9 separate cases. Seven of these are relatively uninteresting; we show only two as examples. If the last rule used was $R\wedge$, then the left premiss ends with:

$$\frac{\Gamma \Rightarrow E \downarrow j \quad \Gamma \Rightarrow F \downarrow k}{\Gamma \Rightarrow E \wedge F \downarrow j + k + 1}$$

where $n = j + k + 1$. The right premiss is then $\Gamma \oplus E \wedge F \Rightarrow C \downarrow m$. Because $L\wedge$ for **G3ip** is invertible, there exists some $m' \leq m$ such that:

$$\Gamma, E, F \Rightarrow C \downarrow m'$$

The following derivation, with two uses of the induction hypothesis, completes this case:

$$\dfrac{\Gamma \Rightarrow F \downarrow k \qquad \dfrac{\dfrac{\Gamma \Rightarrow E \downarrow j}{\Gamma, F \Rightarrow E \downarrow j}\, w \qquad \Gamma, E, F \Rightarrow C \downarrow m'}{\exists d'.\ \Gamma, F \Rightarrow C \downarrow d'}\, ih}{\exists d.\ \Gamma \Rightarrow C \downarrow d}\, ih$$

This is formalised as follows:

case (*ConjR* Γ' *E j F k*)
with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **have** $\Gamma \oplus E \wedge * F \Rightarrow C \downarrow m$ **by** *simp*

then obtain m' **where** $\Gamma \oplus E \oplus F \Rightarrow C \downarrow m'$ **using** *inversionConjL* **by** *auto*
from ‹$\Gamma' \Rightarrow E \downarrow j$› **and** ‹$\Gamma$=$\Gamma'$› **have** $\Gamma \oplus F \Rightarrow E \downarrow j$ **using** *dpWeak* **by** *auto*
with ‹$\Gamma \oplus E \oplus F \Rightarrow C \downarrow m'$› **and** *IH*
  and ‹x= *length* A› **and** ‹$A = E \wedge * F$›
  have $\exists\ n.\ \Gamma \oplus F \Rightarrow C \downarrow n$ **by** (*auto simp add*:*union-ac*)
with ‹$\Gamma' \Rightarrow F \downarrow k$› **and** ‹$\Gamma = \Gamma'$› **and** *IH*
  and ‹x = *length* A› **and** ‹$A = E \wedge * F$›
show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*

It is possible to apply the induction hypothesis twice because F has shorter length than $E \wedge F$. Thus, it does not matter what the depth of the derivation is.

In the next highlighted case, that of $L\wedge$, we use the sub-induction hypothesis. That is, the length of the cut formula is held constant and the depth of the instance of *Cut* is reduced. When the last rule used in the derivation of the left premiss was $L\wedge$, there must have been an occurrence of $E \wedge F$ in Γ. Thus, there is some Γ' such that $\Gamma = \Gamma' \oplus E \wedge F$. The left premiss is then:

$$\frac{\Gamma', E, F \Rightarrow A \downarrow j}{\Gamma', E \wedge F \Rightarrow A \downarrow j+1}$$

where $n = j + 1$. As before, the right premiss of the instance of *Cut* can be rewritten as $\Gamma' \oplus E \wedge F \oplus A \Rightarrow C \downarrow m$. Using the invertibility of $L\wedge$, there is an $m' \leq m$ such that:

$$\Gamma', E, F, A \Rightarrow C \downarrow m'$$

Then, the following derivation, with one application of the induction hypothesis, suffices to remove A:

$$\frac{\dfrac{\Gamma', E, F \Rightarrow A \downarrow j \quad \Gamma', E, F, A \Rightarrow C \downarrow m'}{\exists d'.\ \Gamma', E, F \Rightarrow C \downarrow d'}\ ih}{\exists d.\ \Gamma', E \wedge F \Rightarrow C \downarrow d}$$

This is formalised as follows:

case (*ConjL* Γ' E F A' j)
with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **and** ‹$A = A'$› **have** $\Gamma' \oplus E \wedge * F \oplus A' \Rightarrow C \downarrow m$ **by** *simp*
then obtain m' **where** $m' \leq m$ **and** $\Gamma' \oplus E \oplus F \oplus A' \Rightarrow C \downarrow m'$
  using *inversionConjL* **by** (*auto simp add*:*union-ac*)
with ‹$\Gamma' \oplus E \oplus F \Rightarrow A' \downarrow j$› **have** $\exists\ n.\ \Gamma' \oplus E \oplus F \Rightarrow C \downarrow n$
  using *IH*
  and ‹$A = A'$› **and** ‹x = *length* A› **and** ‹$y = n+m+1$› **and** ‹n=$j+1$› **by** *auto*
then have $\exists\ n.\ \Gamma' \oplus E \wedge * F \Rightarrow C \downarrow n$ **using** *provable-dp.ConjL* **by** *auto*
with ‹$\Gamma = \Gamma' \oplus E \wedge * F$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *simp*

The cases for $L\bot$, $R\vee_1$, $R\vee_2$, $L\vee$ and $L\supset$ are similarly straightforward. This leaves two cases:

where the last rule used was an axiom or an instance of $R\supset$. That the latter requires special attention is owing to $L\supset$ being only partially invertible.

For the former, there is some propositional variable (say At i) and Γ' such that the left premiss is $\Gamma'\oplus\text{At } i \Rightarrow \text{At } i \downarrow 0$. Using this information, the right premiss is $\Gamma'\oplus\text{At } i\oplus\text{At } i \Rightarrow C \downarrow m$. Case analysis on this premiss is then used. If the last rule used on the right was a logical rule (i.e. not an axiom or instance of $L\bot$) then it is easy to show that At i can be cut in the premiss(es) of such a rule. For instance, we show the case of $R\supset$:

$$\dfrac{\dfrac{\Gamma', \text{At } i, E \Rightarrow \text{At } i \downarrow 0 \quad \Gamma', \text{At } i, \text{At } i, E \Rightarrow F \downarrow j}{\exists d'.\ \Gamma', \text{At } i, E \Rightarrow F \downarrow d'}}{\exists d.\ \Gamma', \text{At } i \Rightarrow E \supset F \downarrow d}$$

This is formalised as follows:

case (*Ax i* Γ′)
with ‹Γ ⊕ *A* ⇒ *C* ↓ *m*› **have** Γ ⊕ *At i* ⇒ *C* ↓ *m* **by** *simp*
then show ∃ *k*. Γ ⇒ *C* ↓ *k*
  proof (*cases*)
  case (*ImpR* Γ*1 E F j*)
  from ‹*A* = *At i*› **and** ‹*n*=*0*› **and** ‹Γ ⇒ *A* ↓ *n*› **have** Γ ⇒ *At i* ↓ *0* **by** *simp*
  then have Γ ⊕ *E* ⇒ *At i* ↓ *0* **using** *dpWeak* **by** *auto*
  moreover from ‹Γ*1* ⊕ *E* ⇒ *F* ↓ *j*› **and** ‹Γ ⊕ *At i* = Γ*1*›
    have Γ ⊕ *E* ⊕ *At i* ⇒ *F* ↓ *j* **by** (*auto simp add:union-ac*)
  ultimately have ∃ *n*. Γ ⊕ *E* ⇒ *F* ↓ *n* **using** *IH*
    and ‹*m* = *j*+*1*› **and** ‹*x* = *length A*›
    and ‹*y* = *n*+*m*+*1*› **and** ‹*A* = *At i*› **by** *auto*
  then have ∃ *n*. Γ ⇒ *E*⊃*F* ↓ *n* **using** *provable-dp.ImpR* **by** *auto*
  then show ∃ *n*. Γ ⇒ *C* ↓ *n* **using** ‹*C* = *E* ⊃ *F*› **by** *simp*

Note there is a slight difference between the informal and formal proofs. In the formal proof, we use *Weakening* to introduce the formula E, whereas in the informal proof we give a new axiom with E in the context. This difference is not significant.

If the last rule on the right was an axiom, then we do slightly different things depending on whether the propositional atom on the right is the same as the one on the left. However, both cases are fairly straightforward, and are omitted.

For the final case, where the last rule used on the left was $R\supset$, we again have to perform case analysis on the rule used to derive the right premiss. For the right rule, we cut in the premiss(es). As an example, consider $R\vee_1$:

$$\dfrac{\dfrac{\Gamma \Rightarrow E \supset F \downarrow j+1 \quad \Gamma, E \supset F \Rightarrow G \downarrow k}{\exists d'.\ \Gamma \Rightarrow G \downarrow d'}}{\exists d.\ \Gamma \Rightarrow G \vee H \downarrow d}$$

If the last rule used was a left rule (other than $L\supset$), then we use the corresponding inversion

result on the left premiss (not shown). The case where the last rule used was $L\supset$ is the only remaining case. There are two further subcases: one where the cut formula is the principal formula of the right premiss, and one where it is not. The latter is shown briefly, the former in detail. For the former, the usual transformation (see proof of theorem 4 on page 16) is used and is formalised as follows:

case (*ImpL* $\Gamma 1\ G\ H\ k\ C'\ l$)
have $E \supset F = G \supset H \vee E \supset F \neq G \supset H$ **by** *blast*
moreover
  {**assume** $E \supset F \neq G \supset H$ — Details omitted
   have $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
  }
moreover
 {**assume** $E \supset F = G \supset H$
  then have $E = G$ **and** $F = H$ **by** *auto*
  with $\langle \Gamma' \oplus E \supset F = \Gamma 1 \oplus G \supset H \rangle$ **have** $\Gamma' = \Gamma 1$ **by** *auto*
  with $\langle \Gamma = \Gamma' \rangle$ **and** $\langle \Gamma' \oplus E \Rightarrow F \downarrow j \rangle$ **and** $\langle E = G \rangle$ **and** $\langle F = H \rangle$
    and $\langle \Gamma \Rightarrow A \downarrow n \rangle$ **and** $\langle A = E \supset F \rangle$
    have $\Gamma 1 \oplus G \Rightarrow H \downarrow j$ **and** $\Gamma 1 \Rightarrow G \supset H \downarrow n$ **by** *auto*
  with $\langle \Gamma 1 \oplus G \supset H \Rightarrow G \downarrow k \rangle$ **have** $\exists\ n.\ \Gamma 1 \Rightarrow G \downarrow n$
    using *IH*
    and $\langle x = \mathit{length}\ A \rangle$ **and** $\langle A = E \supset F \rangle$ **and** $\langle E \supset F = G \supset H \rangle$
    and $\langle y = n+m+1 \rangle$ **and** $\langle m = k+l+1 \rangle$ **by** *auto*
  then obtain n' **where** $\Gamma 1 \Rightarrow G \downarrow n'$ **by** *blast*
  with $\langle \Gamma 1 \oplus G \Rightarrow H \downarrow j \rangle$ **have** $\exists\ n.\ \Gamma 1 \Rightarrow H \downarrow n$
    using *IH*
    and $\langle x = \mathit{length}\ A \rangle$ **and** $\langle A = E \supset F \rangle$ **and** $\langle E \supset F = G \supset H \rangle$ **by** *auto*
  then obtain m' **where** $\Gamma 1 \Rightarrow H \downarrow m'$ **by** *blast*
  with $\langle \Gamma 1 \oplus H \Rightarrow C' \downarrow l \rangle$ **have** $\exists\ n.\ \Gamma 1 \Rightarrow C' \downarrow n$
    using *IH*
    and $\langle x = \mathit{length}\ A \rangle$ **and** $\langle A = E \supset F \rangle$ **and** $\langle E \supset F = G \supset H \rangle$ **by** *auto*
  with $\langle \Gamma' = \Gamma 1 \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** $\langle C = C' \rangle$
    have $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
 }
ultimately show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *blast*
qed — This completes the proof

Using this result, it is easy to show that context-splitting *Cut* is admissible:

lemma *contextSplitCut*:
 assumes $\Gamma \Rightarrow A \downarrow n$ **and** $\Gamma' \oplus A \Rightarrow C \downarrow m$
 shows $\exists\ a.\ \Gamma + \Gamma' \Rightarrow C \downarrow a$
proof−
 from *assms* **have** $\Gamma + \Gamma' \Rightarrow A \downarrow n$ **using** *dpWeak'* **by** *auto*
 moreover

from *assms* **have** $\Gamma' \oplus A + \Gamma \Rightarrow C \downarrow m$ **using** *dpWeak'* **by** *auto*
then have $\Gamma + \Gamma' \oplus A \Rightarrow C \downarrow m$ **by** (*simp only*:*union-ac*)
ultimately
show $\exists\ a.\ \Gamma + \Gamma' \Rightarrow C \downarrow a$ **using** *cutAdmissibility* **by** (*auto simp add*:*union-ac*)
qed

4.4.3 Mechanisation Statistics

The entire file, including all auxiliary results, is 840 lines long (contained in appendix D). The proof of *Cut* admissibility is 314 lines long and consists of 570 proof steps. The proofs of invertibility (of which there are three) are 323 lines long and contain 594 proof steps. Although each is a very similar proof, we nevertheless have to prove them all separately. It is undesirable that such seemingly trivial lemmata account for more effort, in terms of writing and proof obligations, than a non-trivial result such as *Cut* admissibility.

4.5 Formalising Contraction Admissibility

G4ip is a calculus for intuitionistic logic with the property that root-first search terminates without loop detection [Dyckhoff and Negri, 2000]. A more complete description of the calculus, including a formalisation of the rules with height, is given in section 4.2 and appendix A. In this section, we give a detailed formalisation of [Dyckhoff and Negri, 2000] up to, and including, the proof of *Contraction* admissibility. As the authors of [Dyckhoff and Negri, 2000] note, *Cut* admissibility for **G4ip** can be proved indirectly by proving the calculus is equivalent to **G3ip**; we showed in section 4.4 that *Cut* admissibility holds for **G3ip**. Thus, we do not give a direct proof of *Cut* admissibility for **G4ip**.

The induction measure used is that outlined in section 4.4.1. However, the length of a formula is *not* the first entry in the pair, rather we use the *weight* of the formula. This is defined as follows:

consts *weight* :: *form* ⇒ *nat*
primrec
weight (*At i*) = *1*
weight (*A* ⊃ *B*) = *1* + *weight A* + *weight B*
weight (*A* ∧∗ *B*) = *2* + *weight A* + *weight B*
weight (*A* ∨∗ *B*) = *3* + *weight A* + *weight B*
weight (*ff*) = *0*

4.5.1 Auxiliary Results

The rules of **G4ip** are more complex than those of **G3ip**. They are also more numerous. Whereas all *Cut* admissibility for **G3ip** needed was *Weakening* and invertibility, here we require further results before *Contraction* can be shown admissible. Before these extra results, however, we note the following invertibility lemmata which are *used*[4]:

Lemma 1 (Invertibility of G4ip [Dyckhoff and Negri, 2000]) *The following rules are height-preserving admissible in* ***G4ip****:*

$$\frac{\Gamma, A \vee B \Rightarrow E}{\Gamma, A \Rightarrow E}\ L\vee_{1,inv} \qquad \frac{\Gamma, A \vee B \Rightarrow E}{\Gamma, B \Rightarrow E}\ L\vee_{2,inv}$$

$$\frac{\Gamma, A \wedge B \Rightarrow E}{\Gamma, A, B \Rightarrow E}\ L\wedge_{inv} \qquad \frac{\Gamma \Rightarrow A \supset B}{\Gamma, A \Rightarrow B}\ R\supset_{inv}$$

$$\frac{\Gamma, At\ i \supset B \Rightarrow E}{\Gamma, B \Rightarrow E}\ L0\supset_{inv} \qquad \frac{\Gamma, (C \wedge D) \supset B \Rightarrow E}{\Gamma, C \supset (D \supset B) \Rightarrow E}\ L\wedge\supset_{inv}$$

$$\frac{\Gamma, (C \vee D) \supset B \Rightarrow E}{\Gamma, C \supset B, D \supset B \Rightarrow E}\ L\vee\supset_{inv} \qquad \frac{\Gamma, (C \supset D) \supset B \Rightarrow E}{\Gamma, B \Rightarrow E}\ L\supset\supset_{inv}$$

The proofs are standard inductions.

As for the non-standard lemmata, firstly we show that generalised axioms are admissible[5]:

$$\Gamma, A \Rightarrow A$$

for arbitrary A. This is done by an strong induction on the weight of A:

```
lemma genAx:
  assumes w = weight A
  shows ∃ n. Γ ⊕ A ⇒ A ↓ n
  using assms
proof (induct w arbitrary: A rule:nat-less-induct)
```

Where A is a propositional atom, $\bot$, a conjunction or a disjunction, the result is quite simple. Where the main connective is an implication, we perform case analysis on the antecedent:

```
case (Imp E F)
then show ∃ n. Γ ⊕ A ⇒ A ↓ n using ‹A=E⊃F›
    proof (cases E)
```

[4] $R\wedge$ is invertible, but this is not required for the formal proofs which follow.

[5] Other authors call them expanded axioms e.g. [Ciabattoni and Terui, 2006a].

This gives five cases:

- $A = \text{At}\ i \supset F$
- $A = \bot \supset F$
- $A = (G \wedge H) \supset F$
- $A = (G \vee H) \supset F$
- $A = (G \supset H) \supset F$

Only four of these cases are treated in [Dyckhoff and Negri, 2000], although the missing case ($A = \bot \supset F$) is trivial. We show the most complicated case, where $A = (G \wedge H) \supset F$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, G \supset (H \supset F) \Rightarrow G \supset (H \supset F)}\, IH}{\Gamma, G \supset (H \supset F), G \Rightarrow H \supset F}\, inv}{\Gamma, G \supset (H \supset F), G, H \Rightarrow F}\, inv}{\Gamma, (G \wedge H) \supset F, G, H \Rightarrow F}\, L{\wedge}{\supset}}{\Gamma, (G \wedge H) \supset F, G \wedge H \Rightarrow F}\, L{\wedge}}{\Gamma, (G \wedge H) \supset F \Rightarrow (G \wedge H) \supset F}\, R{\supset}$$

As noted in [Dyckhoff and Negri, 2000], the induction hypothesis applies because $w(G \supset (H \supset F)) < w((G \wedge H) \supset F)$. This is formalised as follows:

case (*Conj G H*)
then have *weight* $(G \supset (H \supset F)) <$ *weight A* **using** ‹$A = E \supset F$› **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G \supset (H \supset F) \Rightarrow G \supset (H \supset F) \downarrow n$ **using** *IH* **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G \supset (H \supset F) \oplus G \Rightarrow H \supset F \downarrow n$ **using** *inversionImpR* **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G \supset (H \supset F) \oplus G \oplus H \Rightarrow F \downarrow n$ **using** *inversionImpR* **by** *auto*
then obtain *n* **where** *a*: $\Gamma \oplus G \supset (H \supset F) \oplus G \oplus H \Rightarrow F \downarrow n$ **by** *blast*
from *a* **have** $\Gamma \oplus (G \wedge * H) \supset F \oplus G \oplus H \Rightarrow F \downarrow n{+}1$ **using** *provable.ImpLC*
  by (*auto simp add*:*union-ac*)
then have $\Gamma \oplus (G \wedge * H) \supset F \oplus G \wedge * H \Rightarrow F \downarrow n{+}2$
  using *provable.ConjL* **by** (*auto simp add*:*union-ac*)
then have $\Gamma \oplus (G \wedge * H) \supset F \Rightarrow (G \wedge * H) \supset F \downarrow n{+}2{+}1$
  using *provable.ImpR* **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** *auto*

That there is a case not covered is surprising. However, it shows the usefulness of a formalisation, since it was formalisation which uncovered the missing case. Had this result not been formalised the gap, however small, may have remained:

case *ff*
have $\Gamma \oplus ff \supset F \oplus ff \Rightarrow F \downarrow 0$ **by** *auto*

then have $\Gamma \oplus \mathit{ff} \supset F \Rightarrow \mathit{ff} \supset F \downarrow 1$ **using** *provable.ImpR* **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **by** *auto*

Using this lemma, it is easy to show that modus ponens is derivable:

$$\frac{\Gamma, A \supset B \Rightarrow A \supset B}{\Gamma, A, A \supset B \Rightarrow B}\ R\supset_{inv}$$

The next auxiliary lemma required is the following:

Lemma 2 *The rule:*

$$\frac{\Gamma \Rightarrow D \quad \Gamma, B \Rightarrow E}{\Gamma, D \supset B \Rightarrow E}$$

is admissible in ***G4ip****.*

Proof. Induction on the height n of the derivation of the left premiss. If $n = 0$, there are two subcases. In the first, $\bot \in \Gamma$, and so the conclusion is an instance of $L\bot$. In the second case, there is some propositional atom At i such that $D =$ At i and At $i \in \Gamma$. We can then apply $L0\supset$ to the second premiss.

When $n > 0$ there are several cases:

1. The last inference is an invertible left inference: $L\wedge, L\vee, L\wedge\supset, L\vee\supset, L0\supset$. Then, we apply the appropriate inversion lemma to the right premiss, the induction hypothesis, and then the appropriate rule.

2. The last inference is a right rule: $R\wedge, R\vee, R\supset$. In each case, a combination of the induction hypothesis on the premiss(es), *Weakening* and the appropriate rule from $L\wedge\supset, L\vee\supset, L\supset\supset$ is used.

3. The last inference was an instance of $L\supset\supset$. Then, $\Gamma = \Gamma' \oplus (F \supset G) \supset H$ and the premisses are $\Gamma' \oplus G \supset H \oplus F \Rightarrow G$ and $\Gamma' \oplus H \Rightarrow E$. The following derivation suffices:

$$\dfrac{\dfrac{\Gamma', G\supset H, F \Rightarrow G}{\Gamma', G\supset H, F, E\supset B \Rightarrow G}\ w \qquad \dfrac{\Gamma', H \Rightarrow E \qquad \dfrac{\Gamma', (F\supset G)\supset H, B \Rightarrow S}{\Gamma', H, B \Rightarrow S}\ inv}{\Gamma', H, E\supset B \Rightarrow S}\ ih}{\Gamma', (F\supset G)\supset H, E\supset B \Rightarrow S}\ L\supset\supset$$

⊣

The formalisation of this case is as follows:

case (*ImpLL* Γ' F G H n E m)
then obtain a **where** $\Gamma' \oplus (F\supset G)\supset H \oplus B \Rightarrow S \downarrow a$ **by** *blast*
then have $\exists\ m.\ \Gamma' \oplus H \oplus B \Rightarrow S \downarrow m$ **using** *inversionImpLL* **by** (*auto*)

then have *a*: $\exists\, a.\ \Gamma' \oplus H \oplus E{\supset}B \Rightarrow S \downarrow a$ **using** *prems*(6)[**where** $B{=}B$] **by** *auto*

moreover — Left hand derivation

from ‹$\Gamma' \oplus F \oplus G{\supset}H \Rightarrow G \downarrow n$› **have** $\Gamma' \oplus G{\supset}H \oplus F \oplus E{\supset}B \Rightarrow G \downarrow n$
  using *dpWeak* **by** (*auto*)
ultimately show *?case* **using** *provable.ImpLL* **by** (*auto simp add*:*union-ac*)

The final auxiliary lemma we need is one which is somewhat counter-intuitive:

Lemma 3 (Double $D \supset B$) *The rule:*

$$\frac{\Gamma, (C \supset D) \supset B \Rightarrow E}{\Gamma, C, D \supset B, D \supset B \Rightarrow E}$$

is admissible in ***G4ip****.*

Given that we are trying to show *Contraction* is admissible, showing this seems unhelpful. However, $D \supset B$ is a lighter formula than $(C \supset D) \supset B$, so we can use the manipulation to create two lighter formulae. In the proof, when $(C \supset D) \supset B$ is non-principal, the result is straightforward. The proof is by induction on the height of the derivation of the premiss, and the interesting case is where the last inference has $(C \supset D) \supset B$ principal. Then, the premisses are $\Gamma \oplus C \oplus D \supset B \Rightarrow D$ and $\Gamma \oplus B \Rightarrow E$ and the following is a valid derivation:

$$\dfrac{\dfrac{\Gamma, C, D \supset B \Rightarrow D \quad \dfrac{\Gamma, B \Rightarrow E}{\Gamma, C, D \supset B, B \Rightarrow E}\, w}{\Gamma, C, D \supset B, D \supset B \Rightarrow E}\ \text{Lemma 2}}{}$$

This is formalised as follows:

lemma *twoDB*:
  assumes $\Gamma \oplus (C{\supset}D){\supset}B \Rightarrow E \downarrow n$
  shows $\exists\ m.\ \Gamma \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow m$
  using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma \oplus (C{\supset}D){\supset}B\ E\ n$ *arbitrary*: Γ)

case (*ImpLL* $\Gamma\ F\ G\ H\ n\ E\ m\ \Gamma'$)
have $(F{\supset}G){\supset}H = (C{\supset}D){\supset}B \vee (F{\supset}G){\supset}H \neq (C{\supset}D){\supset}B$ **by** *blast*

moreover
 {**assume** $(F{\supset}G){\supset}H = (C{\supset}D){\supset}B$
  then have *eqs*: $F{=}C$
    $G{=}D$
    $H{=}B$
    $\Gamma{=}\Gamma'$ **using** *prems* **by** *auto*
  from *prems* **have** $\Gamma \oplus H \oplus F \oplus G{\supset}H \Rightarrow E \downarrow m$ **using** *dpWeak'* **by** *auto*

```
  moreover
  have Γ ⊕ F ⊕ G⊃H ⇒ G ↓ n by fact
  ultimately
  have ∃ n. Γ ⊕ F ⊕ G⊃H ⊕ G⊃H ⇒ E ↓ n using ImpLClassical
       by (auto simp add:union-ac)
  then have ∃ n. Γ' ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ n using eqs by blast
 }
moreover
 {assume (F⊃G)⊃H ≠ (C⊃D)⊃B — Non-principal case. Proof omitted
  have ∃ n. Γ' ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ n by blast
 }
ultimately
show ?case by blast
```

4.5.2 A Proof of Contraction Admissibility for G4ip

As is now standard, we intersperse the informal proof of the lemma with the formalised version.

Lemma 4 *The rule:*

$$\frac{\Gamma, A, A \Rightarrow E}{\Gamma, A \Rightarrow E}$$

is admissible in ***G4ip****.*

Proof. The proof is by induction on the lexicographic order of the weight of A and the depth of the derivation of the premiss:

```
lemma contract:
fixes A :: form
  and n :: nat
assumes Γ ⊕ A ⊕ A ⇒ C ↓ n
shows ∃ k. Γ ⊕ A ⇒ C ↓ k
using assms
proof (induct x ≡ weight A y ≡ n  arbitrary: Γ A C n rule: nat-prod-induct)
  case (less x y)
  then have IH: ⋀ B m C Γ.
              ⟦(weight B, m) <* (weight A, n); Γ ⊕ B ⊕ B ⇒ C ↓ m⟧ ⟹
              ∃ a. Γ ⊕ B ⇒ C ↓ a by simp
```

If A is $\bot$, then the conclusion is an axiom. If A is some propositional atom, say At i, then we do case analysis on the last rule used:

case (*At i*)
then have $\Gamma \oplus At\ i \oplus At\ i \Rightarrow C \downarrow n$ **using** *prems* **by** *auto*
then have $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$
  proof (*cases*)

In the cases where the last rule used was an axiom or $L\bot$, then the conclusion is likewise an axiom or $L\bot$. If the last rule used was a logical rule, then apply the induction hypothesis to the premiss(es) and then apply the rule. As an example, consider the case of $L\wedge\supset$:

$$\dfrac{\dfrac{\Gamma', \mathrm{At}\ i, \mathrm{At}\ i, E \supset (F \supset G) \Rightarrow D}{\Gamma', \mathrm{At}\ i, E \supset (F \supset G) \Rightarrow D}\ ih}{\Gamma', \mathrm{At}\ i, (E \wedge F) \supset G \Rightarrow D}\ L\wedge\supset$$

where $\Gamma = \Gamma' \oplus \mathrm{At}\ i$. This is formalised as follows:

case (*ImpLC* Γ' *E F G D s*)
from ‹$\Gamma \oplus At\ i \oplus At\ i = \Gamma' \oplus (E \wedge * F) \supset G$›
  obtain $\Gamma 1$
  where *eq1*: $\Gamma = \Gamma 1 \oplus (E \wedge * F) \supset G$
  and *eq2*: $\Gamma' = \Gamma 1 \oplus At\ i \oplus At\ i$
  by *auto*
from *eq2* **and** ‹$\Gamma' \oplus E \supset (F \supset G) \Rightarrow D \downarrow s$›
  have $\Gamma 1 \oplus At\ i \oplus At\ i \oplus E \supset (F \supset G) \Rightarrow D \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus At\ i \oplus E \supset (F \supset G) \Rightarrow D \downarrow k$
  using ‹*A*=*At i*› **and** ‹*n*=*s*+1› **and** *IH* **by** (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma 1 \oplus At\ i \oplus (E \wedge * F) \supset G \Rightarrow D \downarrow k$ **using** *provable.ImpLC* **by** *auto*
then show $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$ **using** ‹*C*=*D*› **and** *eq1* **by** (*auto simp add*:*union-ac*)

When A is compound, we again consider the last rule used. If A is of the form $C \wedge D, C \vee D, \mathrm{At}\ i \supset B, (C \wedge D) \supset B$ or $(C \vee D) \supset B$ and is principal in the last inference, then we use inversion on the other occurrence of A, the induction hypothesis on the lighter formulae, and then the rule again. As an example, consider $L\vee$. Then, $A = S \vee T$ and the premisses are $\Gamma \oplus S \vee T \oplus S \Rightarrow C$ and $\Gamma \oplus S \vee T \oplus T \Rightarrow C$. The appropriate derivation is then:

$$\dfrac{\dfrac{\dfrac{\Gamma, S \vee T, S \Rightarrow C}{\Gamma, S, S \Rightarrow C}\ inv}{\Gamma, S \Rightarrow C}\ ih \qquad \dfrac{\dfrac{\Gamma, S \vee T, T \Rightarrow C}{\Gamma, T, T \Rightarrow C}\ inv}{\Gamma, T \Rightarrow C}\ ih}{\Gamma, S \vee T \Rightarrow C}\ L\vee$$

Where A is non-principal, an application of the induction hypothesis in the premiss(es) will suffice. Suppose the last inference (still for $A = S \vee T$) was $L\wedge$, with principal formula $E \wedge F$:

$$\dfrac{\dfrac{\Gamma, E, F, S \vee T, S \vee T \Rightarrow C}{\Gamma, E, F, S \vee T \Rightarrow C}\ ih}{\Gamma, E \wedge F, S \vee T \Rightarrow C}\ L\wedge$$

These two are formalised as follows:

case (*Disj S T*)
then have $\Gamma \oplus S \vee * T \oplus S \vee * T \Rightarrow C \downarrow n$ **using** *prems* **by** *auto*
then have $\exists\ k.\ \Gamma \oplus S \vee * T \Rightarrow C \downarrow k$
  proof (*cases*)

  case (*ConjL* Γ' *E F D s*)
  from ‹$\Gamma \oplus S \vee * T \oplus S \vee * T = \Gamma' \oplus E \wedge * F$› **obtain** $\Gamma 1$
      where *eq1*: $\Gamma = \Gamma 1 \oplus E \wedge * F$
      and *eq2*: $\Gamma' = \Gamma 1 \oplus S \vee * T \oplus S \vee * T$
      by *auto*
  from *eq2* **and** ‹$\Gamma' \oplus E \oplus F \Rightarrow D \downarrow s$›
     have $\Gamma 1 \oplus S \vee * T \oplus S \vee * T \oplus E \oplus F \Rightarrow D \downarrow s$ **by** *simp*
  then have $\exists\ k.\ \Gamma 1 \oplus S \vee * T \oplus E \oplus F \Rightarrow D \downarrow k$
     using ‹$A{=}S \vee * T$› **and** ‹$n{=}s{+}1$› **and** *IH* **by** (*auto simp add*:*union-ac*)
  then have $\exists\ k.\ \Gamma 1 \oplus S \vee * T \oplus E \wedge * F \Rightarrow D \downarrow k$ **using** *provable.ConjL* **by** *auto*
  then show $\exists\ k.\ \Gamma \oplus S \vee * T \Rightarrow C \downarrow k$ **using** ‹$C{=}D$› **and** *eq1* **by** (*auto*)

  case (*DisjL* Γ' *E D s F t*)
  assume $S \vee * T = E \vee * F$
   then have $\Gamma \oplus E \vee * F = \Gamma'$ **using** ‹$\Gamma \oplus S \vee * T \oplus S \vee * T = \Gamma' \oplus E \vee * F$› **by** *simp*
   from ‹$\Gamma' \oplus E \Rightarrow D \downarrow s$› **and** ‹$\Gamma \oplus E \vee * F = \Gamma'$›
       have $\Gamma \oplus E \vee * F \oplus E \Rightarrow D \downarrow s$ **by** *simp*
   then have $\exists\ k.\ \Gamma \oplus E \oplus E \Rightarrow D \downarrow k$ **using** *inversionDisjL*
       by (*auto simp add*:*union-ac*)
   then have $\exists\ k.\ \Gamma \oplus E \Rightarrow D \downarrow k$ **using** ‹$S \vee * T{=}E \vee * F$› **and** ‹$A{=}S \vee * T$›
       and *IH* **by** *auto*

   moreover — Right hand derivation
   from ‹$\Gamma' \oplus F \Rightarrow D \downarrow t$› **and** ‹$\Gamma \oplus E \vee * F = \Gamma'$›
       have $\Gamma \oplus E \vee * F \oplus F \Rightarrow D \downarrow t$ **by** *simp*
   then have $\exists\ k.\ \Gamma \oplus F \oplus F \Rightarrow D \downarrow k$ **using** *inversionDisjL*
       by (*auto simp add*:*union-ac*)
   then have $\exists\ k.\ \Gamma \oplus F \Rightarrow D \downarrow k$ **using** ‹$S \vee * T{=}E \vee * F$› **and** ‹$A{=}S \vee * T$›
       and *IH* **by** *auto*
   ultimately show $\exists\ k.\ \Gamma \oplus S \vee * T \Rightarrow C \downarrow k$ **by** *blast*

The most interesting case is when $A = (V \supset W) \supset T$ and is principal for the last inference. Then, we use lemma 3 in the following derivation, where $ih^\star$ is three applications of the induction hypothesis:

$$\dfrac{\dfrac{\dfrac{\Gamma, W \supset T, (V \supset W) \supset T, V \Rightarrow W}{\Gamma, W \supset T, W \supset T, W \supset T, V, V \Rightarrow W}\ \text{Lemma 3}}{\Gamma, W \supset T, V \Rightarrow W}\ ih^{\star} \qquad \dfrac{\dfrac{\Gamma, (V \supset W) \supset T, T \Rightarrow D}{\Gamma, T, T \Rightarrow D}\ inv}{\Gamma, T \Rightarrow D}\ ih}{\Gamma, (V \supset W) \supset T \Rightarrow D}\ L{\supset}{\supset}$$

The induction hypothesis may be used three times since every time it is an instance of *Contraction* on a lighter formula. The formalisation is similar:

```
case (ImpLL Γ' E F G s D t)
then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
  proof (cases S)
  case (Imp V W)
  have (V⊃W)⊃T = (E⊃F)⊃G ∨ (V⊃W)⊃T ≠ (E⊃F)⊃G by blast
  moreover
   {assume (V⊃W)⊃T = (E⊃F)⊃G
    then have Γ ⊕ (V⊃W)⊃T = Γ'
        using ‹S=V⊃W›
        and ‹Γ ⊕S⊃T⊕S⊃T = Γ' ⊕ (E⊃F)⊃G› by simp
    then have Γ ⊕ (V⊃W)⊃T ⊕ V ⊕ W⊃T ⇒ W ↓ s
        using ‹(V⊃W)⊃T = (E⊃F)⊃G›
        and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s› by auto
    then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⊕ W⊃T ⊕ W⊃T ⇒ W ↓ k
        using twoDB by (auto simp add:union-ac)
    then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⊕ W⊃T ⇒ W ↓ k
        using ‹A=S⊃T› and ‹S=V⊃W›
        and IH by auto
    then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⇒ W ↓ k using ‹A=S⊃T› and ‹S=V⊃W›
        and IH by auto
    then have ∃ k. Γ ⊕ V ⊕ W⊃T ⇒ W ↓ k using ‹A=S⊃T› and ‹S=V⊃W›
        and IH by (auto simp add:union-ac)

    moreover — Right hand derivation

    from ‹Γ ⊕ (V⊃W)⊃T = Γ'› and ‹Γ' ⊕ G ⇒ D ↓ t›
        have Γ⊕(V⊃W)⊃T ⊕ T ⇒ D ↓ t using ‹(V⊃W)⊃T = (E⊃F)⊃G›
        by auto
    then have ∃ k. Γ ⊕ T ⊕ T ⇒ D ↓ k using inversionImpLL
        by (auto simp add:union-ac)
    then have ∃ k. Γ ⊕ T ⇒ D ↓ k using ‹A=S⊃T› and IH by auto
    ultimately
    have ∃ k. Γ ⊕ (V⊃W)⊃T ⇒ D ↓ k using provable.ImpLL by auto
    then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ‹S=V⊃W› and ‹C=D› by simp
```

```
  }
moreover
  {assume (V⊃W)⊃T ≠ (E⊃F)⊃G — Non-principal case omitted
   have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k  by (auto)
  }
ultimately show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by blast
```

All cases have been covered, and the proof is complete. ⊣

4.5.3 Mechanisation Statistics

The proof of *Contraction* admissibility is 917 lines long, consisting of 1867 proof steps (see appendix D). The proof of *Weakening* admissibility is 57 lines long, consisting of 108 proof steps. The proofs of invertibility are 953 lines long, consisting of:

- 271 proof steps for $L\wedge$,
- 335 proof steps for $L\vee$,
- 274 proof steps for $L0\supset$,
- 271 proof steps for $L\wedge\supset$,
- 271 proof steps for $L\vee\supset$,
- 271 proof steps for $L\supset\supset$ and
- 145 proof steps for $R\supset$

This is a total of 1838 proof steps. Thus, the standard lemmata of invertibility and *Weakening* admissibility account for more lines and more proof steps than the proof of *Contraction* admissibility. This was also true in section 4.4.

4.6 Conclusions

We have given three novel, non-trivial results from Structural Proof Theory. They have been formalised in the proof assistant *Isabelle* using the human-readable variant *Isar*. Whilst they may not be fully readable by someone unfamiliar with the syntax of *Isar*, the proofs are certainly more readable than a tactic script. As such, the proofs can be made to closely mirror the informal pen-and-paper approach. In the case of *Contraction* (section 4.5), the formalisation procedure highlighted where an informal proof was (very slightly) incomplete.

The first-order binding package *Nominal Isabelle* was used in section 4.3. We exploited the straightforward definition of substitution possible in *Nominal*, allowing us to concentrate on the proof of Craig's interpolation theorem rather than minor issues about substitution.

The proofs are also relatively succinct. Obviously, they are not as short as some text-book proofs, where many cases are treated by giving a general method and an invitation given to the reader to verify the details. However, one can be sure that a checked proof is correct. The onerous part of the process was the requirement to prove invertibility lemmata (in section 4.4 and section 4.5) and *Weakening*. These are all standard induction proofs which justifiably have informal proofs consisting of the line "Trivial" or "Standard induction." Using *Isabelle*, however, one has no recourse to this option. Thus, the standard lemmata accounted for roughly half of the entire length of the file in section 4.4, and not much less in section 4.5. They certainly had a roughly similar length (both in terms of the number of lines and the number of proof steps required) to the main result of their respective section.

Thus, the natural question becomes: is it possible to get *Isabelle* to prove these results itself? Or rather, is there some result one could find which, when formalised, changes 300 lines of (user-written) proof into 30 lines of proof? It is this question which will occupy us in the following chapter.

Chapter 5

Classifying Sequent Calculi

5.1 Introduction

In the previous chapter, the formalised proof of *Cut* admissibility was quite short, whereas the invertibility lemmata were quite long and tedious to prove. It would be helpful, therefore, to give results which would ensure that the rules of a calculus are invertible, thus precluding the need to prove invertibility on a case-by-case, or calculus-by-calculus, basis. In order to do this effectively, one must reason about classes of calculi, without knowing the specifics of their rules. So that we can do that, in this chapter we provide a framework into which various sequent calculi fit. A few such frameworks exist in the literature. Some are more appropriate for our purposes than others. They were developed with the principle aim of giving sufficient or necessary conditions for a calculus so that it would admit *Cut*. Since *Cut* admissibility is of a different nature to invertibility, we find that the approaches are not wholly suited to the study of invertibility. Therefore, we give our own, new framework under which, given appropriate restrictions, invertibility results are easy to prove.

Attempts to codify sequent calculi can be rather abstract. To this end, a wide variety of examples are given to bring the framework to life.

5.1.1 Structure of the chapter

In section 5.2, we examine the *canonical propositional calculi* of [Avron and Lev, 2001]. This was further extended to handle first-order logics in [Zamansky and Avron, 2006]. The *simple* sequent calculi of [Ciabattoni and Terui, 2006a] are analysed in section 5.3. In section 5.4, we look at the framework found in [Restall, 1999]. These three sections contain material from the literature. In section 5.5, we outline our own, novel approach.

Section 5.6 gives a proof using our framework that instances of *Cut* which are non-principal can be permuted into the premisses of the left-hand derivation. Finally, in section 5.7 we discuss why axioms are restricted to propositional variables. These sections contain

new material.

5.1.2 Notation

Recall from chapter 4 that we use the symbol $\oplus$ and $+$ to stand for the sequent calculus comma (for single formulae and structures of formulae respectively). So:

$$\Gamma, \Gamma' \Rightarrow \Delta, A$$

would be displayed as $\Gamma + \Gamma' \Rightarrow \Delta \oplus A$.

Clearly, this abbreviation is context-sensitive. Γ and Γ' could be lists and the comma operation "append," or they could be sets and the comma operation set-theoretic union. It will be obvious from the context what the symbols represent.

5.2 Canonical Calculi

In this section, we will look closely at the canonical calculi of [Avron and Lev, 2001]. Canonical calculi are similar to what we will develop in section 5.5. We also see three methods for manipulating a rule set, two of which will be useful for later sections.

In order to closely follow the terminology of [Avron and Lev, 2001], we will refer to a sequent calculus as a *Gentzen-type system*. This concept is introduced without definition. Formulae are given by $\phi, \psi, \ldots$, whilst *sets* of formulae $(\Gamma, \Delta, \ldots)$ are used as contexts. The use of sets for contexts means that both *Contraction* and *Exchange* are implicit. We have the following:

Definition 2 (Standard [Avron and Lev, 2001]) *A Gentzen-type system G is **standard** if its set of axioms includes the standard axioms $\Gamma \oplus \psi \Rightarrow \Delta \oplus \psi$ and it has all the standard structural rules (including Cut).* ⊣

The use of *standard* axioms will mean that *Weakening* is likely admissible.[1] Since *Exchange* and *Contraction* are implicit, it is assumed in this thesis that the standard structural rules (a concept which is never defined) consist of, but are not necessarily restricted to mingle (**min**):

$$\frac{\Gamma, \Theta \Rightarrow \Delta \quad \Gamma, \Xi \Rightarrow \Delta}{\Gamma, \Theta, \Xi \Rightarrow \Delta}\ \text{min}$$

The use of a generalised axiom, where ψ is *not* restricted to atomic formulae, is harmful to invertibility. The discussion of this will be delayed until section 5.7.

[1] We say "likely" since some other conditions need to be satisfied, too, for *Weakening* to be admissible. This will be addressed in section 7.

A *clause* is a sequent which consists of atomic formulae only. *Canonical rules* are designed so that the calculus obeys the subformula property: the only formulae appearing in the premisses of a rule are subformulae of those formulae in the conclusion of the rule. The rules are also defined to be *pure*. A rule is pure if there are no side conditions limiting its application [Avron and Lev, 2001],[Avron, 1991]. These rules are defined as follows:

Definition 3 (Canonical Rules [Avron and Lev, 2001]) *A **canonical rule** of arity n is an expression of the form:*

$$\frac{\Pi_1 \Rightarrow \Sigma_1 \quad \cdots \quad \Pi_m \Rightarrow \Sigma_m}{C}$$

where $m \geq 0$, C is either $\star(p_1, \ldots, p_n) \Rightarrow \emptyset$ or $\emptyset \Rightarrow \star(p_1, \ldots, p_n)$ for some connective $\star$ (of arity n), and for all $1 \leq i \leq m$, $\Pi_i \Rightarrow \Sigma_i$ is a clause such that $\Pi_i, \Sigma_i \subseteq \{p_1, \ldots, p_n\}$.

*An **application of a canonical rule**:*

$$\frac{\Pi_1 \Rightarrow \Sigma_1 \quad \cdots \quad \Pi_m \Rightarrow \Sigma_m}{\star(p_1, \ldots, p_n) \Rightarrow}$$

is any inference step of the form:

$$\frac{\Gamma_1, \Pi_1^* \Rightarrow \Delta_1, \Sigma_1^* \quad \cdots \quad \Gamma_m, \Pi_m^* \Rightarrow \Delta_m, \Sigma_m^*}{\Gamma, \star(\psi_1, \ldots, \psi_n) \Rightarrow \Delta}$$

where Π_i^* and Σ_i^* are obtained from Π_i and Σ_i (respectively) by substituting ψ_j for p_j (for all $1 \leq j \leq n$), Γ_i, Δ_i are sets of formulae, $\Gamma = \bigcup_{i=1}^m \Gamma_i$, and $\Delta = \bigcup_{i=1}^m \Delta_i$. An application of a canonical rule with a conclusion of the form $\emptyset \Rightarrow \star(p_1, \ldots, p_n)$ is defined similarly. $\dashv$

The condition $\Pi_i, \Sigma_i \subseteq \{p_1, \ldots, p_n\}$ means that canonical rules have the subformula property.

This definition permits *context-splitting* calculi. Such calculi are generally not invertible. For instance, every right rule in a fully invertible calculi like **G3cp** become non-invertible when altered to be context-splitting. We therefore restrict ourselves to context-sharing calculi. In other words, calculi in which $\Gamma_i = \Gamma$ and $\Delta_i = \Delta$ for $1 \leq i \leq m$.

Note that **G3ip** and **G4ip** contain some *non-canonical rules*. Consider the rule for $L\supset$ from **G3ip**, given in the format from the definition above:

$$\frac{p_1 \supset p_2 \Rightarrow p_1 \quad p_2 \Rightarrow}{p_1 \supset p_2 \Rightarrow}$$

The left-hand premiss is *not* a clause; it contains the non-atomic formula $p_1 \supset p_2$.

Consider the rules $L\supset\supset$ and $L0\supset$ from **G4ip**:

$$\frac{p_1, p_2 \supset p_3 \Rightarrow p_2 \quad p_3 \Rightarrow}{(p_1 \supset p_2) \supset p_3 \Rightarrow} \; L\supset\supset \qquad \frac{p_1, p_2 \Rightarrow}{p_1, p_1 \supset p_2 \Rightarrow} \; L0\supset$$

The left-hand rule is non-canonical for the same reason as $L\supset$ was non-canonical. The right-hand rule is non-canonical because it is *not* pure. The side-condition for application of the rule is that p_1 must be atomic. Furthermore, the conclusion of the rule does not contain a single compound formula.

Definition 4 (Canonical Calculus [Avron and Lev, 2001]) *A standard calculus is called* ***canonical*** *if in addition to the standard axioms and the standard structural rules it only has canonical logical rules.* ⊣

G3ip and **G4ip** are *not* canonical: they contain non-canonical logical rules. Thus, the range of calculi which can be analysed using the canonical framework is already quite limited.

Avron and Lev seek to restrict some of the rule patterns which are harmful to *Cut* admissibility proofs. They alter a canonical calculus by removing *superfluous* and *redundant* rules:

Definition 5 (Superfluous and Redundant Rules [Avron and Lev, 2001]) *A canonical rule is called* ***superfluous*** *if it is possible to obtain the empty clause from its premisses using cuts.*

A logical rule in a canonical calculus G is called ***redundant in*** *G if its set of premisses is subsumed by*[2] *the set of premisses of another rule of G which has the same conclusion.* ⊣

Example. [Avron and Lev, 2001] Consider a rule which has premisses $p_1 \oplus p_2 \Rightarrow \emptyset$ and $p_1 \Rightarrow p_2$ and $\emptyset \Rightarrow p_1$. Then this rule is superfluous; we can derive the empty clause using cuts:

$$\dfrac{\Rightarrow p_1 \qquad \dfrac{p_1 \Rightarrow p_2 \quad p_1, p_2 \Rightarrow}{p_1 \Rightarrow}}{\Rightarrow}$$

If a calculus contained the two rules:

$$\frac{\Rightarrow p_1}{\star(p_1, p_2) \Rightarrow} \qquad \frac{\Rightarrow p_1 \quad \Rightarrow p_2}{\star(p_1, p_2) \Rightarrow}$$

then the right-hand rule is redundant.

We will see in the next chapter that the fewer rules there are in a calculus (in general), the more likely it is that they will be invertible. Indeed, we will examine this precisely in section 6.5. The first half of the following lemma will be of use, then:

Lemma 5 ([Avron and Lev, 2001]) *Let G by a canonical calculus, and let G' be the calculus that is obtained from G by deleting superfluous and redundant rules. Then G' is equivalent to G. Moreover, every sequent that has a Cut-free proof in G' also has such a proof in G.*

[2] where "is subsumed by" is taken to be "includes".

Proof. We only show the first half of the statement.

Suppose G had a superfluous rule. Then, the empty sequent can be derived from it using cuts. A single instance of *Weakening* can therefore show that the rule is admissible in G'.

Suppose G had the rules:

$$\frac{\Pi_1 \Rightarrow \Sigma_1 \quad \cdots \quad \Pi_{n-1} \Rightarrow \Sigma_{n-1}}{\Rightarrow \star(p_1, \ldots, p_m)}\ r_1$$

$$\frac{\Pi_1 \Rightarrow \Sigma_1 \quad \cdots \quad \Pi_{n-1} \Rightarrow \Sigma_{n-1} \quad \Pi_n \Rightarrow \Sigma_n}{\Rightarrow \star(p_1, \ldots, p_m)}\ r_2$$

r_2 is redundant in G. Then, G' does not contain r_2. To show equivalence of G and G', we show that any derivation in G can be transformed into a derivation in G' and vice versa. The latter is simple: any derivation in G' *is* a derivation in G.

Suppose we had the following derivation in G ending in an application of r_2:

$$\frac{\Gamma_1, \Pi_1^* \Rightarrow \Delta_1, \Sigma_1^* \quad \cdots \quad \Gamma_{n-1}, \Pi_{n-1}^* \Rightarrow \Delta_{n-1}, \Sigma_{n-1}^* \quad \Gamma_n, \Pi_n^* \overset{\mathcal{D}}{\Rightarrow} \Delta_n, \Sigma_n^*}{\Gamma \Rightarrow \Delta, \star(\psi_1, \ldots, \psi_m)}$$

From the same premisses, we get a derivation in G' by discarding the subderivation $\mathcal{D}$. ⊣

A clause is called a *unit clause* if it is of the form $p_1 \Rightarrow \emptyset$ or $\emptyset \Rightarrow p_1$. The final two concepts introduced are as follows:

Definition 6 (Separated and Full Rules [Avron and Lev, 2001]) *A canonical rule is called* ***separated*** *if all its premisses are unit clauses.*

Suppose C is a sequent which is of the form $\emptyset \Rightarrow \star(p_1, \ldots, p_n)$ or $\star(p_1, \ldots, p_n) \Rightarrow \emptyset$. A separated rule:

$$\frac{\Pi_1 \Rightarrow \Sigma_1 \quad \cdots \quad \Pi_m \Rightarrow \Sigma_m}{C}$$

is called ***full*** *if $m = n$ and for every $1 \leq i \leq n$, $\Pi_i \cup \Sigma_i = \{p_i\}$.* ⊣

Examples. The following three rules are *i*. not separated, *ii*. separated but not full and *iii*. full:

$$\frac{p_1, p_2 \Rightarrow}{p_1 \circ p_2 \Rightarrow}\ i \qquad \frac{p_1 \Rightarrow}{p_1 \circ p_2 \Rightarrow}\ ii \qquad \frac{p_1 \Rightarrow \quad p_2 \Rightarrow}{p_1 \circ p_2 \Rightarrow}\ iii$$

A method is outlined which transforms any canonical calculus into an equivalent calculus whose rules are full, not superfluous and not redundant. The method creates a rule set which is larger than the original rule set. For example, the canonical rule $R\vee$ from **G3cp** is expanded to three full rules:

$$\frac{\Rightarrow p_1, p_2}{\Rightarrow p_1 \vee p_2}\ R\vee$$

$$\frac{\Rightarrow p_1 \quad p_2 \Rightarrow}{\Rightarrow p_1 \vee p_2}\ R\vee_1 \qquad \frac{\Rightarrow p_1 \quad \Rightarrow p_2}{\Rightarrow p_1 \vee p_2}\ R\vee_2 \qquad \frac{p_1 \Rightarrow \quad \Rightarrow p_2}{\Rightarrow p_1 \vee p_2}\ R\vee_3$$

We shall see in later sections that this will cause problems for invertibility.

5.2.1 Conclusions

The framework of canonical calculi has been extended to include first-order rules in [Zamansky and Avron, 2006]. The reason for its creation was to study *Cut* admissibility. Some of the methods and alterations may help prove *Cut* admissibility for such calculi, but they make many rules non-invertible. Moreover, the range of calculi which can be called canonical is somewhat small. The use of sets for contexts, structural rules, generalised axioms and context-splitting rules will also create problems when trying to show invertibility.

5.3 Simple Calculi

In this section we look at the simple calculi of [Ciabattoni and Terui, 2006a]. They are ultimately found to be not useful for our purposes.

In this section we follow the notation of [Ciabattoni and Terui, 2006a]. In particular, $\alpha, \beta, \gamma, \ldots$ are propositional variables, $\star_1, \star_2, \ldots$ are logical connectives (with a given arity). $\Gamma, \Delta, \Pi, \Sigma$ are sequences of formulae, where a formula is defined as follows:

Definition 7 (Formulae [Ciabattoni and Terui, 2006a]) *A **formula** A is either a propositional variable of a **compound** formula of the form $\star(A_1, \ldots, A_m)$, where $A_1, \ldots, A_m$ are formulae.* ⊣

Inference rules are specified using *meta-variables* $X, Y, \ldots$, standing for arbitrary formulae, and sequences $\Theta, \Xi, \Phi, \Psi, \Upsilon$ of meta-variables. Note that these sequences can be empty. The use of sequences means most structural rules will be have to be explicit. The use of sets for contexts in section 5.2 meant that *Contraction* and *Exchange* were implicit; here that is not the case. [Ciabattoni and Terui, 2006a] also restricts calculi to single-succedent calculi. Explicit structural rules mean invertibility will be hard to show, whereas the restriction to single-succedent calculi means the framework will not be applicable to many calculi.

The analysis (both syntactic and semantic[3]) is centred around *simple calculi*:

Definition 8 (Simple Calculi [Ciabattoni and Terui, 2006a]) *A single-conclusion sequent calculi $\mathcal{L}$ is called **simple** whenever $\mathcal{L}$ consists of the identity axiom of the form $X \Rightarrow$*

[3]Semantic methods are not the subject of study of this dissertation.

X, together with: the (multiplicative version of the) Cut rule, structural rules $\{(R_i)\}_{i\in\Lambda_0}$ and for each logical connective $\star$, left logical rules $\{(\star,l)_j\}_{j\in\Lambda_1}$ and right logical rules $\{(\star,r)_k\}_{k\in\Lambda_2}$ ($\Lambda_0,\Lambda_1,\Lambda_2$ can be empty):

$$\frac{\Theta \Rightarrow X \quad \Theta_l, X, \Theta_r \Rightarrow \Xi}{\Theta_l, \Theta, \Theta_r \Rightarrow \Xi}\ Cut \qquad \frac{\Upsilon_1 \Rightarrow \Psi_1 \quad \cdots \quad \Upsilon_n \Rightarrow \Psi_n}{\Theta_l \Rightarrow \Xi}\ R_i$$

$$\frac{\Upsilon_1 \Rightarrow \Psi_1 \quad \cdots \quad \Upsilon_n \Rightarrow \Psi_n}{\Theta_l, \star(\vec{X}), \Theta_r \Rightarrow \Xi}\ (\star,l)_j \qquad \frac{\Upsilon_1 \Rightarrow \Psi_1 \quad \cdots \quad \Upsilon_n \Rightarrow \Psi_n}{\Theta_l \Rightarrow \star(\vec{X})}\ (\star,r)_k$$

In rules $R_i, (\star,l)_j$ and $(\star,r)_k$, $m \geq 0$ and the meta-variables in Θ_l, Θ_r (called left context meta-variables), those in Ξ (called right context meta-variables) and the meta-variables in $\vec{X} = X_1, \ldots, X_m, m \geq 0$ (called active meta-variables) are mutually disjoint. The active meta-variables $X_1, \ldots, X_m$ are mutually distinct. In addition, structural rules satisfy the following condition:

(str) *Any meta-variable in $\Upsilon_1, \ldots, \Upsilon_n$ is a left context meta-variable, and any meta-variable in $\Psi_1, \ldots, \Psi_n$ is a right context meta-variable.*

while logical rules satisfy:

(log0) *Any $\Upsilon_1, \ldots, \Upsilon_n$ is either an active or a left context meta-variable, and any meta-variable in $\Psi_1, \ldots, \Psi_n$ is either an active or a right context meta-variable.*

(log1) *Each meta-variable occurs at most once in Θ_l, Θ_r.*

(log2) *If $(I)[X]$ is a logical rule of $\mathcal{L}$ with a left context meta-variable X, then $(I)[\Phi]$ also belongs to $\mathcal{L}$ for any sequence Φ of fresh and distinct meta-variables. Here, $(I)[\Phi]$ denotes the rule obtain from (I) by replacing all the occurrences of X with Φ.*

(log3) *If $(I)[Y]$ is a logical rule of $\mathcal{L}$ with a right context meta-variable Y, then $(I)[\Phi_l; \Phi_r \Rightarrow \Xi]$ also belongs to $\mathcal{L}$ for any sequent $\Phi_l, \Phi_r \Rightarrow \Xi$ that consists of fresh and distinct meta-variables. Here, $(I)[\Phi_l; \Phi_r \Rightarrow \Xi]$ denotes a logical rule obtained from I by replacing all sequents of the form $\Theta \Rightarrow Y$ with $\Phi_l, \Theta, \Phi_r \Rightarrow \Xi$.*

⊣

These conditions allow for both context-sharing and context-splitting rules. The rules will also satisfy the subformula property, because any meta-variable in Υ_i and Ψ_i will appear in the conclusion as either a left context meta-variable, right context meta-variable or active meta-variable. As with section 5.2, there is no restriction on axioms to be propositional formulae.

An *instance* of a logical or structural rule is obtained by substituting arbitrary formulae for meta-variables. *Context formulae* replace context meta-variables, and so on. The formula of the form $\star(\vec{A})$ as well as the formulae replacing X in identity axioms are called *principal*.

Examples. The calculus $\mathbf{FL}^{\mathbf{n}}_{\mathbf{e_k}}$ for intuitionistic linear logic with knotted structural rules from [Hori et al., 1994] fits into this framework[4]. In particular, the initial sequent $L\bot$ is seen as a 0-premiss rule:

$$\frac{}{\Theta_l, \bot, \Theta_r \Rightarrow \Xi}$$

Similarly, the rules for the logical constants 0 and 1 are given by 0-premiss rules. The right rules for fusion and conjunction are given by:

$$\frac{\Theta_1 \Rightarrow X_1 \quad \Theta_2 \Rightarrow X_2}{\Theta_1, \Theta_2 \Rightarrow X_1 * X_2}\ R* \qquad \frac{\Theta \Rightarrow X_1 \quad \Theta \Rightarrow X_2}{\Theta \Rightarrow X_1 \wedge X_2}\ R\wedge$$

For $R*$, we have $\Upsilon_1 = \Theta_1, \Upsilon_2 = \Theta_2, \Psi_1 = X_1, \Psi_2 = X_2$ and $\Theta_l = \Theta_1 + \Theta_2$, where $+$ is used to concatenate sequences. The active meta-variables are distinct. $R*$ satisfies (log0): no context meta-variables pass from the right to the left of the sequent arrow, and vice versa. The other three conditions can be verified ((log2) does not need to be satisfied for this rule).

The knotted structural rules are generalisations of *Weakening* and *Contraction*, which take n occurrences of a meta-variable X in the premiss to k occurrences in the conclusion:

$$\frac{\Theta_l, \overbrace{X, \ldots, X}^{n}, \Theta_r \Rightarrow \Xi}{\Theta_l, \underbrace{X, \ldots, X}_{k}, \Theta_r \Rightarrow \Xi}\ (n \rightsquigarrow k)$$

In particular, normal *Weakening* is $(0 \rightsquigarrow 1)$ and normal *Contraction* is $(2 \rightsquigarrow 1)$. It is not true that every instance of this knotted structural rule can be placed within the simple framework. For all $n > 0$, the rule $(n \rightsquigarrow 0)$ does not fit into the framework, for it will violate (str). For example:

$$\frac{\Theta_l, X, \Theta_r \Rightarrow \Xi}{\Theta_l, \Theta_r \Rightarrow \Xi}\ (1 \rightsquigarrow 0)$$

The meta-variable X will not be a left context meta-variable since it does not appear in the conclusion of the rule.

Any calculus which has rules requiring multiple formulae to be principal will not fit into this framework. Since the rule $L0\supset$ for **G4ip** has two principal formulae, **G4ip** cannot be analysed under this framework.

[4] In a sense. Once $\mathbf{FL}^{\mathbf{n}}_{\mathbf{e_k}}$ has been altered to use sequences, not multisets, it fits into the simple calculus framework.

5.3.1 Conclusions

The simple framework is used for single-succedent calculi. It contains a lot of elements which, if not restricted, will be harmful to the invertibility of the rules any calculus which fits the framework. Structural rules, the use of sequences for contexts, generalised axioms and the allowing of context-sharing logical rules will cause problems in later sections. That being said, if these elements are removed, it could be possible to create a more amenable framework for studying invertibility. The framework we propose in section 5.5 will share some components with the simple framework.

5.4 Restall's Framework

The final framework we look at is that of Restall [Restall, 1999]. It is more flexible than the others presented. The major difference between our framework, to be developed in section 5.5, is the characterisation of rules as infinite sets of inferences.

In [Restall, 1999], Restall outlines a framework, inspired by [Belnap Jr., 1982], which is more general than those of the previous sections. In particular, sequent calculi are not the only objects which can be studied under the framework. However, the most obvious application is sequent calculi. Since the study of invertibility in sequent calculi is the main aim of this thesis, we will apply his framework to sequent calculi alone. Note that we have also changed the notation from [Restall, 1999] to closely follow the notation used throughout this dissertation.

The structures in a sequent $X \Rightarrow Y$ are not specified. Whilst X and Y consist of formulae, they could be lists, sets, multisets, sequences or single formulae. A formula instance is in either *antecedent* or *consequent* position, but not both. Normally, the notions of antecedent and "on the left of a sequent arrow", and consequent and "on the right side of a sequent arrow" coincide, but this need not be the case. In the case of *display logic* [Belnap Jr., 1982] a formula in antecedent position can occur on the right of the sequent arrow. However, in what follows, we treat antecedent and "on the left" as interchangeable.

Definition 9 (Inferences and Rules [Restall, 1999]) *An **inference** is a (possibly empty) set of sequents, called premisses, with a sequent, called the conclusion.*

*A **rule** is a set of inferences, which may be presented schematically.* ⊣

So, an axiom would be treated as a 0-premiss inference. Note that the definition allows for infinite premiss rules. Here there is a departure from [Belnap Jr., 1982], since there only finitely many premisses for inferences are allowed.

Restall investigates how certain *parameter conditions* affect sequent calculi. Parameters are defined intuitively:

Definition 10 (Parameters, Congruences and Principal [Restall, 1999]) *A **parameter** is something which is held constant from premisses to conclusion or introduced with no*

regard to their particulars.

*The **congruence relation** on parameters is some equivalence relation on parameters in an inference.*

*Nonparametric formula instances in the conclusion of an inference are **principal** for that inference.* ⊣

For instance, Γ and Γ' can be congruent in the following inferences:

$$\frac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow \circ A, \Delta} \qquad \frac{\Gamma' \Rightarrow A, \Delta}{\Gamma' \Rightarrow \circ A, \Delta}$$

Definition 11 (Parameter Conditions [Restall, 1999])

- ***Shape-alikeness of parameters.*** *Congruent parameters are occurrences of the same structure.*
- ***Position-alikeness of parameters.*** *Congruent parameters are either all antecedent or all consequent parts of their respective sequents.*
- ***Non-proliferation of parameters.*** *A congruence class of parameters in an inference contains at most one formula in the conclusion of that inference.*

⊣

These correspond to conditions $C2 - C4$ from [Belnap Jr., 1982].

Examples. Consider the following three inferences:

$$\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A * B, \Delta}\ i \qquad \frac{\Gamma \Rightarrow A, \Delta \quad \Gamma \Rightarrow B, \Delta}{\Delta \Rightarrow A * B, \Gamma}\ ii \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma, \Gamma \Rightarrow \Delta, \Delta, A * B}\ iii$$

In i, all three parameter conditions are satisfied. In ii, the position-alikeness condition is violated, for Γ appears in antecedent position in the premisses and consequent position in the conclusion. In iii, unless the structures Γ and Δ are such that $\Gamma + \Gamma = \Gamma$ (for instance they are sets and $+$ is union), then this inference violates the non-proliferation of parameters condition.

In particular, these conditions will rule out some structural inferences, but not others. *Contraction*, for instance, will satisfy the parameter conditions. Rules from modal logic (see [Troelstra and Schwichtenberg, 2000]), such as:

$$\frac{\Box\Gamma \Rightarrow A, \Diamond\Delta}{\Gamma', \Box\Gamma \Rightarrow \Box A, \Diamond\Delta, \Delta'}\ R\Box$$

may violate the non-proliferation of parameters condition.

As with the other frameworks, we are restricted as to the principal formula of inferences:

Definition 12 (Single Principal Constituents [Restall, 1999]) *If there is an inference ending in a sequent C in which some formula A is principal, then A is the only principal formula in C, unless C is an axiom.* ⊣

This is condition $C5$ in [Belnap Jr., 1982]. Owing to the rule $L0\supset$ from **G4ip** having two principal formulae, **G4ip** will not fit into Restall's framework.

Using these conditions (and some more specific ones which are not covered here), Restall shows that non-principal cuts can be permuted upwards in such a calculus. The details are not important and we shall prove this result using our own framework in section 5.6.

5.4.1 Conclusions

Restall's framework is well-developed and general. Unfortunately, the generality means that a lot of harmful rules are allowable. Moreover, rules which (we shall see) are not harmful to invertibility are not allowed; they fail the parameter conditions. Again, this is understandable; the main purpose of the framework in [Restall, 1999] was to analyse calculi to show when *Cut* is admissible, not when its rules are invertible. However, in [Curry, 1963] (which inspired [Belnap Jr., 1982]), there is some study of invertibility. We will discuss this work in the next chapter, and in particular show the results obtained are weaker versions of our results.

None of the studied frameworks discussed are truly appropriate for our task. Therefore we develop our own framework which takes elements from the other three, but refines them so that we can analyse invertibility in sequent calculi.

5.5 Definitions

We are interested in the occurrences of formulae in inferences. In what follows, when we talk of formulae, metaformulae, ..., we mean *formula occurrences*, *metaformula occurrences*, ... This abbreviation is used to aid readability.

We distinguish between formulae and metaformulae. The reason for this distinction is so that we can distinguish rules from inferences later. Let P, the set of propositional atoms, be defined by the grammar $p, q, \ldots$. Then, a *formula* is defined by the grammar:

$$A ::= P \mid \bot \mid F(A \text{ list})$$

where F ranges over constructors. For instance, if the language included conjunction, an example of a formula could be $(p \wedge q) \wedge \bot \equiv \text{Conj } [\text{Conj } [p, q], \bot]$. A formula is an expression in the object language.

Suppose that $A, B, C \ldots$ are formula variables, and $P, Q, R, \ldots$ are atom variables, then *metaformulae* are given by the following grammar:

$$\phi ::= A \mid P \mid \bot \mid F(\phi \text{ list})$$

An example of a metaformula is $\bot \supset (A \supset P)$, if implication was one of the constructors. We instantiate metaformulae with formulae.

$\Gamma, \Delta, \ldots$ are *metamultisets of formulae*. In other words, $\Gamma, \Delta, \ldots$ range over multisets of formulae.

The use of multisets, instead of lists or sequences, means that the results are applicable for systems in which the structural *Exchange* rule:

$$\frac{\Gamma, B, A, \Gamma' \Rightarrow \Delta}{\Gamma A, B, \Gamma' \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, B, A, \Delta'}{\Gamma \Rightarrow \Delta, A, B, \Delta'}$$

is admissible.

In the informal analysis, we will represent a compound formula as $\star_s(\vec{A})$ or $\circ_t(\vec{B})$, where $\star_s$ is an s-ary connective, and $\vec{A}$ has length s, for example. The usual propositional formulae can be displayed in this manner, for example $A \wedge B$ would be displayed as $\text{Conj}_2\ [A, B]$.

We distinguish between *rules* and instances of rules, which we call *inferences*. Rules are pairs consisting of a list of sequents, called the *premisses*, and a single sequent called the *conclusion*. Most of our lemmata are about rules, and one proves these by showing the result for arbitrary instances of the rule. Inferences can contain *no* metaformulae; they have all been instantiated with formulae. Likewise every metamultiset is instantiated; however we do not highlight the distinction between the metamultisets and their instantiations, for the sake of readability. We denote rules as $R, S, T, \ldots$ and inferences as $r, s, t \ldots$. For example, the inference on the right is an instance of the rule on the left:

$$\frac{\Gamma \Rightarrow \Delta, \phi, \psi}{\Gamma \Rightarrow \Delta, \phi \vee \psi} \qquad \frac{\Gamma \Rightarrow \Delta, p, q}{\Gamma \Rightarrow \Delta, p \vee q}$$

We impose conditions upon a calculus which ensure that the rules are invertible. To define properly these conditions, we require the following definitions about formulae:

Definition 13 (Subformulae, Metasubformulae) *The **subformulae** of a formula A are defined by induction on the structure of A as follows:*

- *A is a subformula of A*
- *If $F[A_1, \ldots, A_n]$ is a subformula of A, then so are $A_1, \ldots, A_n$*

*The **metasubformulae** of a metaformula ϕ are defined by induction on the structure of ϕ in the same fashion.* ⊣

Some logical rules, called context sharing rules, are such that every premiss of a rule has the same context. Each context sharing rule can, therefore, be split into two components.

First, there is the *active* part of a rule: those metaformulae, and metamultisets of formulae, which cannot be arbitrarily instantiated in an inference, and the *passive* part of a rule, which can be arbitrarily instantiated. The latter part is really the context of the rule, in propositional calculi. We make this more precise:

Definition 14 (Active and Passive Metaformulae) *A metaformula occurrence ϕ is **active** for a rule R iff:*

- *ϕ cannot be arbitrarily instantiated in an instance of R, OR*
- *ϕ is a submetaformula occurrence of an active metaformula for R, occurring outside the active metaformula.*

*A metaformula is **passive** for a rule R if it is not active for R.* ⊣

Note we are only defining the active metaformulae for a *rule*, not all occurrences of the metaformula.

The *active part* of a rule is then obtained by deleting all passive metaformulae, and metamultisets of passive formulae, from the rule. Similarly, the *passive part* of a rule is obtained by deleting all active metaformulae from the rule. These concepts are similar to the active and context formulae of [Ciabattoni and Terui, 2006a](section 5.3).

As an example, consider the rules $L\supset$ from **G3cp** and $L0\supset$ from **G4ip**, shown with their active parts:

$$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \supset \psi \Rightarrow \Delta} \qquad \frac{\Rightarrow \phi \quad \psi \Rightarrow}{\phi \supset \psi \Rightarrow} \qquad \frac{\Gamma, P, \phi \Rightarrow \psi}{\Gamma, P, P \supset \phi \Rightarrow \psi} \qquad \frac{P, \phi \Rightarrow}{P, P \supset \phi \Rightarrow}$$

In the former case $\phi \supset \psi$ cannot be instantiated with a conjunctive formula, for instance, and so is active. This, in turn, means ϕ and ψ are active. In the latter case, the same reasoning means $P \supset \phi$ is active, therefore P and ϕ are active. In more complicated rules, the active part of the rule is larger:

$$\frac{\Box\Gamma \Rightarrow \phi, \Diamond\Delta}{\Box\Gamma, \Gamma' \Rightarrow \Box\phi, \Diamond\Delta, \Delta'}$$

It is not possible to instantiate $\Box\Gamma$ with a single formula whose outermost modal operator is $\Diamond$. Therefore, the active part of this rule is:

$$\frac{\Box\Gamma \Rightarrow \phi, \Diamond\Delta}{\Box\Gamma \Rightarrow \Box\phi, \Diamond\Delta}$$

Note that structural rules have *no active metaformulae* according to this definition. As an example, take the *Contraction* rule, where the contracted formula is ϕ:

$$\frac{\Gamma, \phi, \phi \Rightarrow \Delta}{\Gamma, \phi \Rightarrow \Delta}$$

ϕ can be arbitrarily instantiated and so is *passive* by our definition.

This method of giving a rule, and then removing the active parts for analysis, is different from that of [Avron and Lev, 2001] (section 5.2). There, the active parts were given and context was built around them. A discussion of the differences this can create is given in section 7.2.

We focus on a particular kind of rule, one in which certain structural rules (which, in general, are harmful to invertibility if primitive in the calculus), are not allowed.

Definition 15 (Decomposable rule) *We call a rule R **decomposable** iff, after deleting all active metaformulae from R to obtain R', we have:*

$$\frac{\Gamma \Rightarrow \Delta \quad \cdots \quad \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ R' \qquad OR \qquad \frac{\Gamma \Rightarrow \Delta \quad \cdots \quad \Gamma \Rightarrow \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\ R'$$

*If every rule in a calculus $\mathcal{R}$ is decomposable, then we call $\mathcal{R}$ **decomposable**.* ⊣

As an example, the *Contraction* rule is *not decomposable*. For, when we remove the active formulae (of which there are none), we do not have the passive part fitting either of the forms above. These conditions may be verified by eye, checking the rules. We now distinguish between two kinds of decomposable rules.

Definition 16 (Normal, Implicit-Weakening Rules) *A rule R is called **normal** iff after deleting all active metaformulae from R to obtain R', the conclusion of R' is the same as each premiss of R'.*

*A rule R is called an **implicit-weakening** rule (IW rule) iff after deleting all active metaformulae from R to obtain R', the conclusion of R' is not equal to each premiss of R'.* ⊣

Rules whose passive part is like that shown on the left of definition 15 are normal, whereas those whose passive part is like that shown on the right are IW rules.

Note that, in the decomposition of a rule, it could be that a part is a rule in its own right, although this is not necessary. We call the rule an *extension* of the active part. Normal rules will satisfy the parameter conditions from section 5.4 (definition 11, [Restall, 1999]). IW rules *may* violate the non-proliferation of parameters condition.

The decomposition makes it straightforward to identify the *principal metaformulae of a decomposable rule (inference)*; they are any metaformulae (formulae) which appears in the conclusion of the active part of the rule (inference).

We study two families of decomposable rules: *axioms* and *uniprincipal rules*. Axioms have no premisses, and both the antecedent and the succedent of the conclusion must contain some atom variable P, or the antecedent must contain $\bot$. In other words:

$$\frac{}{\Gamma, P \Rightarrow \Delta, P} \qquad \frac{}{\Gamma, \bot \Rightarrow \Delta}$$

are axioms.

Another approach would be to not restrict axioms to atom variables. So, we could use *any* ϕ in place of P in the above formulation, as seen in section 5.2, section 5.3 and section 5.4. This restricted form of axioms limits the number of calculi we can analyse. However, we are interested in characterising invertibility for calculi. The use of unrestricted axioms causes problems for invertibility, as will be discussed in section 5.7.

A *uniprincipal* rule must have a compound formula in the conclusion of its active part, and furthermore this compound formula must be the *only* formula in the conclusion. This restriction does not allow $L0\supset$ from **G4ip**, for instance, to be a uniprincipal rule; the conclusion of the active part was $P\oplus P\supset\phi\Rightarrow\emptyset$. Uniprincipal rules will therefore have *single-principal constituents* ([Restall, 1999], section 5.4), and will fit into both the frameworks of section 5.2 and section 5.3.

For uniprincipal rules, we can also define *principal on the left* and *principal on the right* for a rule (inference) by noting whether the single metaformula (formula) in the conclusion of the active part of the rule (inference) is in the antecedent or succedent, respectively.

For the remainder of this chapter and the next (except section 6.7), we restrict our attention to uniprincipal rules. A sequent calculus is thus defined by some set of uniprincipal rules, joined with the set of axioms. We will usually just talk about the uniprincipal rules defining a calculus, with the understanding that the calculus contains the set of axioms.

We need the notion of *derivability*, and, because we are interested in height-preserving invertibility (in the first instance), we need the notion of *derivability at height* n. Axioms are derivable at height 0, and should every premiss in a rule be derivable at height *at most* n, then the conclusion of the rule will be derivable at height $n+1$.

Definition 17 (Invertible rule) *For a calculus defined by a set of primitive rules* $\mathcal{R}$*, we say that a rule* $R\in\mathcal{R}$ *with premisses* $P_1,P_2,\ldots,P_n$ *and conclusion* C*, is* ***(height preserving) invertible with respect to*** $\mathcal{R}$ *if, for each premiss* P_i*, the rule:*

$$\frac{C}{P_i}$$

is (height preserving) admissible.

If every such $R\in\mathcal{R}$ *is invertible, we say that* $\mathcal{R}$ *is invertible.* ⊣

All of the height preserving admissibility results we prove require the height preserving admissibility of *Weakening* (this is more commonly known as depth-preserving *Weakening*). We give a lemma in the next chapter that guarantees that *Weakening* is depth-preserving, but require the following definition to do so:

Definition 18 (Context Dependent Rules [Negri, 2005],[Rasga, 2007]) *A* ***context dependent rule*** *is a rule which has side conditions which place restrictions upon the passive formulae allowable for instantiations of that rule.* ⊣

Many rules with variable binding are context dependent rules, as is the usual rule for necessitation on the right in modal logic:

$$\frac{\Box\Gamma \Rightarrow \phi, \Diamond\Delta}{\Box\Gamma, \Gamma' \Rightarrow \Box\phi, \Diamond\Delta, \Delta'}$$

where here the side condition is that all formulae in the antecedent must be boxed.

5.6 Non-Principal Cuts and Invertible Rule Sets

Invertible sets of rules have a nice property with regards to *Cut* admissibility. If the *Cut* formula is not principal in the left premiss of an instance of *Cut* and all rules of a calculus are invertible, we can permute the instance of *Cut* into the premisses of the left-hand derivation. More formally:

Lemma 6 *(Non-Principal Cut Permutation) Suppose $\mathcal{R}$ is an invertible set of decomposable uniprincipal rules. Let:*

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1 & & \mathcal{D}_n \\ \Gamma_1 \Rightarrow \Delta_1 & \cdots & \Gamma_n \Rightarrow \Delta_n\end{matrix}}{\Gamma \Rightarrow \Delta, \phi}\, R \qquad \begin{matrix}\mathcal{D} \\ \phi, \Gamma \Rightarrow \Delta\end{matrix}}{\Gamma \Rightarrow \Delta}\, Cut$$

be a derivation in $\mathcal{R}$ augmented with the Cut rule, denoted $\mathcal{R}^{cut}$. If ϕ is not principal on the right for R, then there exists, for $i = 1, \ldots, n$, $\mathcal{D}'_i$, $\bar{\Gamma}_i$ and $\bar{\Delta}_i$ constructed from $\mathcal{D}$, Γ_i and Δ_i respectively, such that:

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1 & & \mathcal{D}'_1 \\ \bar{\Gamma}_1 \Rightarrow \bar{\Delta}_1, \phi & & \phi, \bar{\Gamma}_1 \Rightarrow \bar{\Delta}_1\end{matrix}}{\bar{\Gamma}_1 \Rightarrow \bar{\Delta}_1} \quad \cdots \quad \dfrac{\begin{matrix}\mathcal{D}_n & & \mathcal{D}'_n \\ \bar{\Gamma}_n \Rightarrow \bar{\Delta}_n, \phi & & \phi, \bar{\Gamma}_n \Rightarrow \bar{\Delta}_n\end{matrix}}{\bar{\Gamma}_n \Rightarrow \bar{\Delta}_n}}{\Gamma \Rightarrow \Delta}\, R$$

is a derivation in $\mathcal{R}^{cut}$.

Proof. ϕ is not principal for R. Suppose R is a normal right rule (the left case is simpler) with principal formula ψ. Then, R is of the form:

$$\frac{\begin{matrix}\Gamma', \Gamma'_1 \Rightarrow \Delta', \Delta'_1 & \ldots & \Gamma', \Gamma'_n \Rightarrow \Delta', \Delta'_n\end{matrix}}{\Gamma' \Rightarrow \Delta', \psi}\, R$$

where $\Gamma'_i \Rightarrow \Delta'_i$ are the active parts of R.

Comparing the two (equivalent) conclusions of R yields $\Gamma = \Gamma'$ and $\Delta \oplus \phi = \Delta' \oplus \psi$. From the latter, there exists $\bar{\Delta}$ such that:

1. $\Delta = \bar{\Delta}, \psi$
2. $\Delta' = \bar{\Delta}, \phi$

We can rewrite the derivations ending in R with 2:

$$\frac{\begin{matrix}\mathcal{D}_1 & & \mathcal{D}_n \\ \Gamma, \Gamma'_1 \Rightarrow \bar{\Delta}, \Delta'_1, \phi & \cdots & \Gamma, \Gamma'_n \Rightarrow \bar{\Delta}, \Delta'_n, \phi\end{matrix}}{\Gamma \Rightarrow \bar{\Delta}, \psi, \phi}\ R$$

In other words, ϕ appears on the right in every premiss of R.

We can rewrite the right derivation in the instance of *Cut* with 1 to give the root as $\Gamma \oplus \phi \Rightarrow \bar{\Delta} \oplus \psi$. Because $\mathcal{R}$ is invertible, R is invertible. Thus, taking the derivation $\mathcal{D}$, we can apply the inversion of R to obtain n derivations corresponding to the n premisses of R. Label these derivations $\mathcal{D}'_i$, for every $i = 1, \ldots, n$. Then, the following are derivable:

$$\begin{matrix}\mathcal{D}'_i \\ \phi, \Gamma, \Gamma'_i \Rightarrow \bar{\Delta}, \Delta'_i\end{matrix}$$

For each i, we cut ϕ:

$$\frac{\begin{matrix}\mathcal{D}_i & & \mathcal{D}'_i \\ \Gamma, \Gamma'_i \Rightarrow \bar{\Delta}, \Delta'_i, \phi & & \phi, \Gamma, \Gamma'_i \Rightarrow \bar{\Delta}, \Delta'_i\end{matrix}}{\Gamma, \Gamma'_i \Rightarrow \bar{\Delta}, \Delta'_i}\ Cut$$

We can then apply R:

$$\frac{\Gamma, \Gamma'_1 \Rightarrow \bar{\Delta}, \Delta'_1 \quad \cdots \quad \Gamma, \Gamma'_n \Rightarrow \bar{\Delta}, \Delta'_n}{\Gamma \Rightarrow \bar{\Delta}, \psi}\ R$$

Rewriting using $\Delta = \bar{\Delta} \oplus \psi$ completes the proof. Thus, we have:

1. $\bar{\Gamma}_i = \Gamma, \Gamma'_i$
2. $\bar{\Delta}_i = \bar{\Delta}, \Delta'_i$

as required. Where R was an IW rule a similar transformation applies. Indeed, where the *Cut* formula ϕ is part of the implicit weakening, the situation is much simpler. ⊣

If the rule R from the lemma is assumed to be height preserving invertible, then the permutation upwards of the instance of *Cut* means the many instances of *Cut* occur at a lower height than the original *Cut*. Thus, when a proof of *Cut* admissibility uses an induction measure based on height, we will be able to apply the induction hypothesis to remove these instances of *Cut*.

5.7 Generalised Axioms are Harmful to Invertibility

Throughout the previous sections, it was stated that allowing *any* formula to be part of an axiom was a bad idea; a restriction to atom variables was applied.

Suppose the set of axioms for **G3ip** were allowed to contain *any* formula. Then:

$$\Gamma, A \wedge B \Rightarrow A \wedge B$$

is derivable at height 0. For the rule $R\wedge$ to be height preserving invertible, we require that $\Gamma \oplus A \wedge B \Rightarrow A$ is derivable at height 0. However, the smallest derivation of this sequent has height 1:

$$\frac{\Gamma, A, B \Rightarrow A}{\Gamma, A \wedge B \Rightarrow A}\ L\wedge$$

Another option is to assume that such generalised axioms are *not* derivable at height 0, but rather that the following are admissible rules for all connectives:

$$\frac{\Gamma, \Lambda \Rightarrow \Delta, \Lambda}{\Gamma, \star_s(\vec{\Phi}) \Rightarrow \Delta, \star_s(\vec{\Phi})}$$

where Λ contains all of the proper subformulae of $\vec{\Phi}$.

It is another area of study as to when such generalised axioms are admissible in a calculus. In [Ciabattoni and Terui, 2006a], this problem is known as *axiom expansion*. Such a problem does not form part of this thesis, however.

5.8 Conclusions

Three frameworks from the literature were studied and found to be inappropriate for our needs. Thus, the relevant ideas were taken from each to create a new framework for the analysis of sequent calculi. Harmful rules such as any context-splitting rules are outlawed. The framework is robust enough that more than just invertibility can be investigated; a step in a proof of *Cut* admissibility was also demonstrated.

A range of calculi fit into this framework, and we can easily analyse when rules in such calculi will be invertible, as we shall see in the next chapter.

Chapter 6

Invertibility

6.1 Introduction

In the last chapter, we developed a new framework under which we could classify and study sequent calculi. In this chapter, we will examine conditions under which various calculi can be said to contain invertible rules. These conditions and the results based upon them, are a major contribution of this thesis.

In section 6.2, the most basic class of multisuccedent rules are analysed. Following that, we specialise the result to single succedent calculi (section 6.3). Rounding off the first set of results, we look at modal logics in section 6.4.

Thereafter, three extensions are studied. In section 6.5, we examine whether some results from the previous chapter can aid invertibility. Finding that one result is particularly unhelpful motivates a new method of manipulating the rules of a calculus, which is explained in section 6.6. In section 6.7, the restriction to uniprincipal rules is relaxed. The invertibility results of [Dawson, 2008] and [Curry, 1963] are proved in section 6.8, as an easy corollary of our lemmata.

Each proof of invertibility relies on the admissibility of dp-*Weakening*. Thus, some sections have explicit proofs that *Weakening* is height preserving admissible.

6.2 Multisuccedent Calculi

Multisuccedent calculi are the best place to start. Symmetry will mean we only have to prove the result for right rules; that the equivalent lemma holds for left-rules is immediate.

We have the following, simple result:

Lemma 7 (dp-Weakening) *Let $\mathcal{R}$ be a calculus defined by a set of uniprincipal rules containing* ***no*** *context-dependent rules. Then, Weakening is height preserving admissible in $\mathcal{R}$. That is, the rule:*

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$.

Proof. Suppose $\Gamma \Rightarrow \Delta$ is an instance of the premiss. The proof is by induction on the height, n, of the derivation of $\Gamma \Rightarrow \Delta$.

If $n = 0$, then the premiss was an axiom. So, there exists some p such that $p \in \Gamma$ and $p \in \Delta$, or $\bot \in \Gamma$. In the first case, $p \in \Gamma + \Gamma'$ and $p \in \Delta + \Delta'$, whilst in the second, $\bot \in \Gamma + \Gamma'$. Since there are no context-dependent rules, we have no side conditions which limit the number of formula occurrences in rules, thus in either case $\Gamma + \Gamma' \Rightarrow \Delta + \Delta'$ will be derivable at height 0.

If $n > 0$, then suppose the last inference in the derivation was r (based on some rule R):

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_n \Rightarrow \Delta_n}{\Gamma \Rightarrow \Delta}\ r$$

There are two subcases: r is either a normal inference, or r is an IW inference.

In the former, we can apply the induction hypothesis to each premiss, to obtain:

$$\Gamma', \Gamma_i \Rightarrow \Delta', \Delta_i \ \text{ for } i = 1, \ldots, n$$

as derivable at height at most $n - 1$. Using these premisses, apply a new inference r', which is likewise based on R, but has a different passive part to r.

In the latter case, we do not need the induction hypothesis. We simply apply a new IW inference r' based on R which contains Γ' and Δ', which completes the proof. ⊣

That there are *no* context dependent rules is very restrictive. In particular, it rules out lots of first-order, and modal, rules. This restriction can be relaxed in later sections. At those times, the *Weakening* result will be extended.

Next, we are going to prove invertibility results for calculi fitting into our framework. We give a set of conditions, which can be adapted for either left or right rules. After the proof, we will give examples of rules which satisfy the conditions. The form of the lemmata may seem odd; however, the additional multisets of formulae in the premisses come from the active part of the rule.

Lemma 8 (Right Multisuccedent Rules) *Let $\mathcal{R}$ be a set of decomposable rules defined by a set of uniprincipal rules. Then, the rule:*

$$\frac{\Gamma \Rightarrow \star_s(\vec{\phi}), \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$ if $\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of every rule with $\star_s(\vec{\phi})$ principal on the right.

Proof. Let $\Gamma \Rightarrow \Delta \oplus \star_s(\vec{B})$ be an instantiation of the premiss. We prove the lemma by induction on the height n of the derivation of $\Gamma \Rightarrow \Delta \oplus \star_s(\vec{B})$. If $n = 0$, then there exists some atom p such that $p \in \Gamma$ and $p \in \Delta \oplus \star_s(\vec{B})$, or $\bot \in \Gamma$. We then have $p \in \Gamma + \Gamma'$ and $p \in \Delta + \Delta'$, or $\bot \in \Gamma + \Gamma'$, and so $\Gamma + \Gamma' \Rightarrow \Delta + \Delta'$ is an axiom.

If $n > 0$, then we do case analysis on the last inference, r, in the derivation of $\Gamma \Rightarrow \Delta \oplus \star_s(\vec{B})$. There are four subcases to consider:

1. r was an instance of a normal rule, and $\star_s(\vec{B})$ is principal for r.
2. r was an instance of an IW rule, and $\star_s(\vec{B})$ is principal for r.
3. r was an instance of a normal rule, and $\star_s(\vec{B})$ is not principal for r.
4. r was an instance of an IW rule, and $\star_s(\vec{B})$ is not principal for r.

Case 1. r is of the form:

$$\frac{\cdots \quad \Gamma, \Gamma' \Rightarrow \Delta, \Delta' \quad \cdots}{\Gamma \Rightarrow \Delta, \star_s(\vec{B})}$$

We have $\Gamma + \Gamma' \Rightarrow \Delta + \Delta'$ is a premiss of r. This means it is derivable at height $n - 1$, and thus at a height not greater than n, as required.

Case 2. r is of the form:

$$\frac{\cdots \quad \Gamma_1, \Gamma' \Rightarrow \Delta_1, \Delta' \quad \cdots}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \star_s(\vec{B})}$$

where Γ_2 and Δ_2 are the implicit weakening parts of the inference. The sequent $\Gamma_1 + \Gamma' \Rightarrow \Delta_1 + \Delta'$ is a premiss of r, and thus is derivable at height $n - 1$. We use lemma 7 to obtain $\Gamma_1 + \Gamma_2 + \Gamma' \Rightarrow \Delta_1 + \Delta_2 + \Delta'$ as derivable at height $n - 1$, but this is just $\Gamma + \Gamma' \Rightarrow \Delta + \Delta'$, and so we are done.

Case 3. There are two further cases here; one where r was an instance of a left rule, and one where r was an instance of a right rule. We show the latter; the former is simpler. Suppose r had principal formula $\circ_t(\vec{D})$, then r is of the form:

$$\frac{\Gamma'', \Gamma''_1 \Rightarrow \Delta'', \Delta''_1 \quad \cdots \quad \Gamma'', \Gamma''_n \Rightarrow \Delta'', \Delta''_n}{\Gamma'' \Rightarrow \circ_t(\vec{D}), \Delta''} \; r$$

Comparing representations of the conclusion, we have $\Gamma \equiv \Gamma''$, and $\Delta \oplus \star_s(\vec{B}) \equiv \Delta'' \oplus \circ_t(\vec{D})$. From this we have a $\Delta^{\sim}$ such that:

1. $\Delta = \Delta^{\sim} \oplus \circ_t(\vec{D})$
2. $\Delta'' = \Delta^{\sim} \oplus \star_s(\vec{B})$

Rewrite r with 2:

$$\frac{\Gamma, \Gamma_1'' \Rightarrow \Delta^{\sim}, \star_s(\vec{B}), \Delta_1'' \quad \cdots \quad \Gamma, \Gamma_n'' \Rightarrow \Delta^{\sim}, \star_s(\vec{B}), \Delta_n''}{\Gamma \Rightarrow \circ_t(\vec{D}), \star_s(\vec{B}), \Delta^{\sim}}\ r$$

So $\star_s(\vec{B})$ is present in *every* premiss of r, and we can therefore apply the induction hypothesis at the lower height of the premisses. Thus, we have:

$$\frac{\dfrac{\Gamma, \Gamma_1'' \Rightarrow \Delta_1'', \Delta^{\sim}, \star_s(\vec{B})}{\Gamma, \Gamma_1'', \Gamma' \Rightarrow \Delta_1'', \Delta^{\sim}, \Delta'} \quad \cdots \quad \dfrac{\Gamma, \Gamma_n'' \Rightarrow \Delta_n'', \Delta^{\sim}, \star_s(\vec{B})}{\Gamma, \Gamma_n'', \Gamma' \Rightarrow \Delta_n'', \Delta^{\sim}, \Delta'}}{\Gamma, \Gamma' \Rightarrow \Delta^{\sim}, \circ_t(\vec{D}), \Delta'}$$

Using the equation 1 we have the result.
Case 4. r is an instance of an IW rule, say R; suppose $\circ_t(\vec{D})$ was principal for r on the right (the left case is similar). Then, r is of the form:

$$\frac{\Gamma_1, \Gamma_1'' \Rightarrow \Delta_1, \Delta_1'' \quad \cdots \quad \Gamma_1, \Gamma_n'' \Rightarrow \Delta_1, \Delta_n''}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \circ_t(\vec{D})}\ r$$

There are two subcases; one where $\star_s(\vec{B})$ is in Δ_1 and another where it is in Δ_2. The proof of the former is almost identical to that of case 3. In the latter, $\star_s(\vec{B})$ was part of the implicit weakening of r.

Thus, there is some $\Delta^{\sim}$ such that $\Delta_2 = \Delta^{\sim} \oplus \star_s(\vec{B})$, and so the conclusion of r is rewritten as:

$$\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta^{\sim}, \star_s(\vec{B}), \circ_t(\vec{D})$$

Taking the premisses of r, apply a new inference, r', based on R, to them; r' is the same as r except that the implicit weakening is $\Gamma_2 + \Gamma'$ on the left and $\Delta^{\sim} + \Delta'$ on the right. We then have:

$$\Gamma_1, \Gamma_2, \Gamma' \Rightarrow \Delta_1, \Delta^{\sim}, \Delta', \circ_t(\vec{D})$$

is derivable at height n. We also know, from comparing the two forms for r, that:

1. $\Gamma_1, \Gamma_2 = \Gamma$
2. $\Delta_1, \Delta_2, \circ_t(\vec{D}) = \Delta, \star_s(\vec{B})$

and so, rewriting the second to include $\Delta^{\sim}$, we have shown:

$$\Gamma, \Gamma' \Rightarrow \Delta, \Delta'$$

is derivable at height n, which completes the case, and the proof. ⊣

The corresponding lemma for invertibility on the left is symmetrical, and so is given without proof:

Lemma 9 (Left Multisuccedent Rules) *Let $\mathcal{R}$ be a set of decomposable rules defined by a set of uniprincipal rules. Then, the rule:*

$$\frac{\Gamma, \star_s(\vec{\phi}) \Rightarrow \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$ if $\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of every rule with $\star_s(\vec{\phi})$ principal on the left.

6.2.1 Examples

Consider the rule $R\wedge$ from **G3cp**, a calculus for classical propositional logic (see appendix A). To show it is invertible is akin to showing the height preserving admissibility of the following two rules:

$$\frac{\Gamma \Rightarrow \phi \wedge \psi, \Delta}{\Gamma \Rightarrow \phi, \Delta}\ R\wedge_{inv,l} \qquad \frac{\Gamma \Rightarrow \phi \wedge \psi, \Delta}{\Gamma \Rightarrow \psi, \Delta}\ R\wedge_{inv,r}$$

To apply lemma 8 to the left rule, take $\Gamma' = \emptyset$ and $\Delta' = \phi$. Then, we need to show that $\emptyset \Rightarrow \phi$ is the active part of a premiss of every rule with $\phi \wedge \psi$ principal on the right. This is by inspection. So, the rule is strongly admissible, and by a similar argument for $R\wedge_{inv,r}$ using lemma 8 we can conclude that $R\wedge$ is invertible in **G3cp**.

Indeed, by similar arguments, all of the rules are invertible, because the following rules are all height preserving admissible:

$$\frac{\Gamma \Rightarrow \phi \wedge \psi, \Delta}{\Gamma \Rightarrow \phi, \Delta}\ R\wedge_{inv,l} \qquad \frac{\Gamma \Rightarrow \phi \wedge \psi, \Delta}{\Gamma \Rightarrow \psi, \Delta}\ R\wedge_{inv,r} \qquad \frac{\Gamma, \phi \wedge \psi \Rightarrow \Delta}{\Gamma, \phi, \psi \Rightarrow \Delta}\ L\wedge_{inv}$$

$$\frac{\Gamma, \phi \vee \psi \Rightarrow \Delta}{\Gamma, \phi \Rightarrow \Delta}\ L\vee_{inv,l} \qquad \frac{\Gamma, \phi \vee \psi \Rightarrow \Delta}{\Gamma, \psi \Rightarrow \Delta}\ L\vee_{inv,r} \qquad \frac{\Gamma \Rightarrow \phi \vee \psi, \Delta}{\Gamma \Rightarrow \phi, \psi, \Delta}\ R\vee_{inv}$$

$$\frac{\Gamma \Rightarrow \phi \supset \psi, \Delta}{\Gamma, \phi \Rightarrow \psi, \Delta}\ R\supset_{inv} \qquad \frac{\Gamma, \phi \supset \psi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \phi}\ L\supset_{inv,l} \qquad \frac{\Gamma, \phi \supset \psi \Rightarrow \Delta}{\Gamma, \psi \Rightarrow \Delta}\ L\supset_{inv,r}$$

The rule for implication on the left for **G3ip** is not invertible (see section 6.3), however the multisuccedent version of the calculus has the following form:

$$\frac{\Gamma, \phi \supset \psi \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \supset \psi \Rightarrow \Delta}$$

This is height preserving invertible; for each premiss, lemma 9 is applicable. For instance, in the left case, $\Gamma' = \phi \supset \psi$ and $\Delta' = \phi$. Furthermore, any rule with $\phi \supset \psi$ principal on the left must be $L\supset$; no other rules have $\phi \supset \psi$ principal on the left.

Note that we talk about *invertibility* of the rules for **G3cp**. Suppose **G3cp** was altered so that $R\wedge$ included an implicit weakening:

$$\frac{\Gamma_1 \Rightarrow \Delta_1, \phi \quad \Gamma_1 \Rightarrow \Delta_1, \psi}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \phi \wedge \psi}\ R\wedge'$$

Then, the lemma would still apply, but with $\Gamma \equiv \Gamma_1 + \Gamma_2$ and $\Delta \equiv \Delta_1 + \Delta_2$:

$$\frac{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \phi \wedge \psi}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \phi} \qquad \frac{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \phi \wedge \psi}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, \psi}$$

In other words, the lemmata of this section would *not* give the invertibility of $R\wedge'$. Rather, a weakened version of each premiss is derivable at a height not greater than the height of the conclusion of $R\wedge'$.

The condition for the applicability of lemmata 8 and 9 is easily verifiable, for a finite set of rules. In fact, for every finite set of rules, the conditions for all of our lemmata are easily verifiable.

6.3 Single succedent calculi

The restriction to single formulae in the succedent of a sequent creates a small problem. We cannot directly apply the results from section 6.2 by simply restricting succedents to contain at most one formula. Recall the definition of a decomposable rule is that all premisses have the same passive part. However, consider the rule $L\supset$ from **G3ip** (see appendix A), here shown with its passive part:

$$\frac{\Gamma, \phi \supset \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \sigma}{\Gamma, \phi \supset \psi \Rightarrow \sigma} \qquad \frac{\Gamma \Rightarrow \quad \Gamma \Rightarrow \sigma}{\Gamma \Rightarrow \sigma}$$

This is easily rectified:

Definition 19 (Single Succedent Decomposable Rule) *A single succedent rule is called* ***single succedent decomposable*** *iff*

1. *The antecedents of the premisses of the passive part are all the same, AND*

2. *Each antecedent of the premisses of the passive part is a submultiset of the antecedent of the conclusion of the passive part, AND*

3. *The succedent of each premisses of the passive part is either empty or the same as the succedent of the conclusion of the passive part.*

⊣

Where confusion cannot arise, we drop "single conclusion" and just talk about decomposable rules.

As an example, i and ii are decomposable, but iii is not:

$$\frac{\Gamma, \phi \supset \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \sigma}{\Gamma, \phi \supset \psi \Rightarrow \sigma}\ i \qquad \frac{\Gamma \Rightarrow \phi \quad \Gamma \Rightarrow \psi}{\Gamma, \Gamma' \Rightarrow \phi \circ \psi}\ ii \qquad \frac{\Gamma, \phi \Rightarrow \sigma \quad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \star \psi}\ iii$$

$$\frac{\Gamma \Rightarrow \quad \Gamma \Rightarrow \sigma}{\Gamma \Rightarrow \sigma}\ i_p \qquad \frac{\Gamma \Rightarrow \quad \Gamma \Rightarrow}{\Gamma, \Gamma' \Rightarrow}\ ii_p \qquad \frac{\Gamma \Rightarrow \sigma \quad \Gamma \Rightarrow}{\Gamma \Rightarrow}\ iii_p$$

The succedent of the left premiss of the passive part of iii (iii_p) is not empty and differs from the conclusion of the passive part, therefore does not satisfy condition 3 from the definition. This rule does not have the subformula property, but this is unrelated to condition 3.

Normal and IW single succedent rules are defined analogously with the multisuccedent rules (section 5.5). We have the restriction that the succedents contain at most one metaformula.

Lemma 10 (Single Succedent Weakening) *Let $\mathcal{R}$ be a set of decomposable uniprincipal rules (containing no context-dependent rules) restricted to single formula succedents. Then, the rules:*

$$\frac{\Gamma \Rightarrow \delta}{\Gamma, \Gamma' \Rightarrow \delta}\ W_L \qquad \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow \delta}\ W_R$$

are height preserving admissible in $\mathcal{R}$.

Proof. A simple induction. ⊣

Lemma 11 (Single Succedent Right and Left Rules) *Let $\mathcal{R}$ be a set of decomposable uniprincipal rules restricted to single formula succedents. Then, the rules:*

$$\frac{\Gamma \Rightarrow \star_s(\vec{\phi})}{\Gamma, \Gamma' \Rightarrow \phi}\ i \qquad \frac{\Gamma, \star_s(\vec{\phi}) \Rightarrow \psi}{\Gamma, \Gamma' \Rightarrow \psi}\ ii$$

are height preserving admissible in $\mathcal{R}$ if i. $\Gamma' \Rightarrow \phi$ (ii. $\Gamma' \Rightarrow \emptyset$) is the active part of a premiss of every rule with $\star_s(\vec{\phi})$ principal on the i. right (ii. left).

Proof. Part *ii* is shown by a very similar proof to that of lemma 9, and is not shown.

For part i, we must be more careful. Again, we perform induction on the height, n, of an instance of the premiss (say $\Gamma \Rightarrow \star_s(\vec{B})$). If $n = 0$, or $n > 0$ and the last inference used was a right inference, the result is immediate.

Suppose the last inference was a left inference, based on R. Then there is some Γ'' and $\circ_t(\vec{D})$ such that:

$$\frac{\Gamma'', \Gamma_1 \Rightarrow D_1 \quad \cdots \quad \Gamma'', \Gamma_n \Rightarrow D_n}{\Gamma'', \circ_t(\vec{D}) \Rightarrow \star_s(\vec{B})}$$

where each D_i is either $\star_s(\vec{B})$ (comes from the premiss having an active part of $\Gamma_i \Rightarrow \emptyset$), or some active formula. In the former case, we can apply the induction hypothesis at the lower height. In the latter, we use depth-preserving *Weakening* to add Γ' to the antecedent. In either case, we get for $i = 1, \ldots, n$:

$$\Gamma', \Gamma'', \Gamma_i \Rightarrow D_i$$

where D_i was either the active formula of the premiss, or B (the instantiation of ϕ). We then apply a new inference based on R with passive part $\Gamma' + \Gamma'' \Rightarrow B$, which completes the proof. ⊣

6.3.1 Examples

Take the standard formulation of **G3ip** from [Troelstra and Schwichtenberg, 2000]. Consider the rules:

$$\frac{\Gamma \Rightarrow \phi}{\Gamma \Rightarrow \phi \vee \psi}\ R\vee_1 \qquad \frac{\Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \vee \psi}\ R\vee_2$$

$$\frac{\Gamma, \phi \Rightarrow \psi}{\Gamma \Rightarrow \phi \supset \psi}\ R\supset \qquad \frac{\Gamma, \phi \supset \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \gamma}{\Gamma, \phi \supset \psi \Rightarrow \gamma}\ L\supset$$

$R\supset$ is invertible; we have the conditions satisfied, because $\phi \supset \psi$ is principal on the right only for this rule, and $\Gamma' = \phi$. Thus, the rule is invertible, as expected.

By contrast, consider the rules for disjunction on the right and implication on the left. Since neither $\Gamma \Rightarrow \psi$ nor $\Gamma \Rightarrow \phi$ is a premiss of *every* rule which has $\phi \vee \psi$ principal on the right, then lemma 11 says nothing about the invertibility of either disjunctive rule, although neither is invertible. Furthermore, the left premiss of $L\supset$ does not satisfy the conditions, owing to it having a different succedent to its conclusion. The rule is partially invertible, however, since the right premiss fulfils the conditions.

6.4 Modal Logics

In order to examine modal calculi, we need to alter the definitions from section 5.5. In particular, we add the notion of modal operators to the definitions of formulae, metaformulae etc. A *formula* is defined by the grammar:

$$A ::= P \mid \bot \mid F(A \text{ list}) \mid G(A \text{ list})$$

where F ranges over propositional constructors, and G ranges over modal constructors. Metaformulae are built then constructed in an analogous way to section 5.5 (pg. 73).

The modal operators G can also act upon metamultisets and multisets. We call a multiset of formulae *modalised* if it is of the form $!\Gamma$, where $!$ is some modal operator.

In addition to Axioms and uniprincipal rules, which were defined for propositional connectives, we give definitions of two kinds of decomposable modal rule. *Basic modal rules* are like uniprincipal rules, except that instead of a single propositional compound metaformula in the conclusion of the active part, there is a single modal metaformula in the conclusion of the active part.

The second kind of decomposable modal rule is a context-dependent rule (recall definition 18 on page 77):

Definition 20 (Modalised Context Rules) *A rule is a* ***modalised context rule*** *iff the active part of the rule contains elements of the form* $!\Gamma$*, where* $!$ *is a modality and* Γ *is a metamultiset.* ⊣

Note that the modalised metamultisets could appear in the antecedent, the succedent, or both.

The rule $L\Diamond$, for a classical calculus extended with the $S4$-modalities [Troelstra and Schwichtenberg, 2000] (see appendix A), is a modalised context rule:

$$\frac{\Box\Gamma, \phi \Rightarrow \Diamond\Delta}{\Box\Gamma, \Diamond\phi, \Gamma' \Rightarrow \Diamond\Delta, \Delta'}\ L\Diamond$$

whereas $R\Diamond$ is not:

$$\frac{\Gamma \Rightarrow \Delta, \phi}{\Gamma \Rightarrow \Delta, \Diamond\phi}\ R\Diamond$$

Modalised context rules have large active parts, and these active parts could be further decomposed into a context, consisting of the modalised metamultisets, and what we will call the *prime part* of the rule. Formally:

Definition 21 (Prime Metaformulae, Prime Part) *A metaformula* ϕ *is* ***prime*** *iff it appears in the active part of a modalised context rule.*

*The **prime part** of a modalised context rule, is the active part of the rule where all modalised metamultisets have been deleted.* ⊣

This definition is extended to prime formulae and instances in the obvious way.

In $L\Diamond$ above, the prime part of the rule is:

$$\frac{\phi \Rightarrow}{\Diamond\phi \Rightarrow}$$

As has been done in previous sections, we prove the height preserving admissibility of *Weakening* for such sequent calculi.

Lemma 12 (Modal dp-Weakening) *Let $\mathcal{R}$ be a calculus containing uniprincipal propositional rules, basic modal rules, and modalised context rules. If every modalised context rule in $\mathcal{R}$ is an IW rule, then the rule:*

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$.

Proof. Let $\Gamma \Rightarrow \Delta$ be an instantiation of the premiss. The proof is by induction on the height n of the derivation of $\Gamma \Rightarrow \Delta$, and is standard. In the case where the last inference is a modalised context inference, we do not use the induction hypothesis, rather apply a new inference with a suitable extension of the conclusion. ⊣

Using the definition of principal formula from section 5.5, where every metaformula in the active part of a conclusion is principal, we can keep the conditions for invertibility very close to those of section 6.2. We must be careful, however; two inferences can differ in the formulae occurring the modalised multisets. Were we to use the same conditions as in section 6.2, we would encounter a problem; $\Box A$ is principal on the left for many instances of the rule $L\Diamond$, given above.

Lemma 13 (Right Modal Rules) *Let $\mathcal{R}$ be a calculus containing uniprincipal propositional rules, basic modal rules, and modalised context rules. The rule:*

$$\frac{\Gamma \Rightarrow !(\vec{B}), \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$ if:

1. *$\Gamma' \Rightarrow \Delta'$ is a premiss of the prime part of every modalised context inference in which $!(\vec{B})$ is principal on the right.*
2. *$\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of every basic modal inference in which $!(\vec{B})$ is principal on the right.*

3. All modalised context rules in $\mathcal{R}$ are IW rules.

Proof. Let $\Gamma \Rightarrow \Delta \oplus !(\vec{B})$ be an instance of the premiss of the above rule. We proceed by induction on the height n of the derivation. If $n = 0$, or $n > 0$ and the last inference was a propositional rule, then we proceed as in the proof of lemma 8 on page 82 (for, $!(\vec{B})$ will never be principal for such an inference). If the last inference r was an instance of a modal rule R, there are four cases:

1. $!(\vec{B})$ is principal for r, and r is a basic modal inference.
2. $!(\vec{B})$ is principal for r, and r is a modalised context inference.
3. $!(\vec{B})$ is non-principal for r, and r is a basic modal inference.
4. $!(\vec{B})$ is non-principal for r, and r is a modalised context inference.

Case 1. The result is immediate, from condition 2.
Case 2. $\Gamma \Rightarrow \Delta \oplus !(\vec{B})$ is the conclusion of a modalised context inference. Let $\bullet_1, \ldots, \bullet_n, !_1, \ldots, !_m$ be modal operators, then for some $\Gamma_1, \ldots, \Gamma_n$, $\Delta_1, \ldots, \Delta_m$ and Γ'', Δ'', we can rewrite r as (using condition 1):

$$\frac{\cdots \quad \bullet_1\Gamma_1, \cdots, \bullet_n\Gamma_n, \Gamma' \Rightarrow !_1\Delta_1, \cdots, !_m\Delta_m, \Delta' \quad \cdots}{\bullet_1\Gamma_1, \cdots, \bullet_n\Gamma_n, \Gamma'' \Rightarrow !(\vec{B}), !_1\Delta_1, \cdots, !_m\Delta_m, \Delta''}$$

From condition 3 and lemma 12, *Weakening* is height preserving admissible; weaken with Γ'' and Δ'' to obtain the desired result.
Case 3. There are two further subcases; one where r is an instance of a normal rule, and one where r is an instance of an IW rule. In the former, we have, from $!(\vec{B})$ being non-principal for r, that $!(\vec{B})$ appears in every premiss of the inference, as part of the context. We can thus apply the induction hypothesis to each premiss. To this set of premisses extended with the new context involving Γ' and Δ', we apply the instance of R which uses that context, and we are done. (The details are the same as in the proof of lemma 8).

In the latter case, again the details are similar to the equivalent case in the proof of 8, and so are omitted.
Case 4. From condition 3, every modalised context rule is an IW rule, and therefore suppose that the root of r was:

$$\bullet_1\Gamma_1, \cdots, \bullet_n\Gamma_n, \Gamma'' \Rightarrow !_\star(\vec{D}), !_1\Delta_1, \cdots, !_m\Delta_m, \Delta''$$

for some modal operators $\bullet_1, \ldots, \bullet_n, !_1, \ldots, !_m$ and multisets $\Gamma_1, \ldots, \Gamma_n, \Delta_1, \ldots, \Delta_m$ and Γ'', Δ''. Because $!(\vec{B})$ is non-principal on the right, then $!(\vec{B}) \in \Delta''$. Therefore, let $\Delta^\sim$ be such that $\Delta'' = \Delta^\sim \oplus !(\vec{B})$. We use a new instance of R which has $\Gamma'' + \Gamma'$ and $\Delta^\sim + \Delta'$ as the context, and we are done. ⊣

Lemma 14 (Left Modal Rules) *Let $\mathcal{R}$ be a calculus containing single-conclusion propositional rules, basic modal rules, and modalised context rules. The rule:*

$$\frac{\Gamma \Rightarrow !(\vec{B}), \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible in $\mathcal{R}$ if:

1. *$\Gamma' \Rightarrow \Delta'$ is a premiss of the prime part of every modalised context inference with $!(\vec{B})$ principal on the left.*
2. *$\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of every basic modal inference with $!(\vec{B})$ principal on the left.*
3. *Every modalised context rule in $\mathcal{R}$ is an IW rule.*

Proof. Symmetric to the proof for right modal rules. ⊣

It should be noted that these are not true invertibility results; we do not reconstruct the premisses of a modal rule exactly, but we derive weakened versions of the premisses of a modal rule (as in section 6.2.1).

6.4.1 Examples

Consider the calculus for classical propositional logic extended with S4-modalities; in other words, **G3cp** together with the four rules for the **S4** modalities $\Box$ and $\Diamond$ (see appendix A for all of the rules of this calculus). The rules for $\Diamond$ were given earlier. The rules for $\Box$ are:

$$\frac{\Box\Gamma \Rightarrow \phi, \Diamond\Delta}{\Box\Gamma, \Gamma' \Rightarrow \Box\phi, \Diamond\Delta, \Delta'}\ R\Box \qquad \frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma, \Box\phi \Rightarrow \Delta}\ L\Box$$

If a derivation in this system had root $\Gamma \Rightarrow \Delta \oplus \Box A$, then we can use lemma 13 to derive $\Gamma \Rightarrow \Delta \oplus A$. For, the only inferences which have $\Box A$ principal on the right are instances of $R\Box$, and:

$$\Rightarrow A$$

is a premiss of the prime part of every such inference.

However, the rule:

$$\frac{\Gamma \Rightarrow \Diamond A, \Delta}{\Gamma \Rightarrow A, \Delta}$$

is not height preserving admissible in this system. We cannot apply lemma 13: $\emptyset \Rightarrow A$ will not be a premiss of the prime part of *every* modalised context inference which has $\Diamond A$

principal on the right. For instance:

$$\frac{\Box\Gamma \Rightarrow B, \Diamond(\Delta, A)}{\Box\Gamma \Rightarrow \Box B, \Diamond(\Delta, A)}$$

has $\Diamond A$ principal on the right: it is in the premiss of the active part of the inference. However, only $\emptyset \Rightarrow B$ is a premiss of the prime part of the inference.

6.5 Superfluous, Redundant and Full Rules

6.5.1 Removal of Superfluous Rules

Recall from section 5.2 that a rule was superfluous if the empty sequent could be derived from its premisses using *Cut*. A (canonical) calculus with superfluous rules was shown to be equivalent to a calculus with all superfluous rules deleted. It is easy to show that the same result holds for calculi which fit into our framework, so long as *Cut* is admissible. What effect will deleting superfluous rules have on invertibility?

Lemma 15 (Removing Superfluous Rules) *Let $\mathcal{R}$ be a set of decomposable uniprincipal rules in which Cut is admissible. Then, there exists a set of decomposable uniprincipal rules $\mathcal{R}'$ which contains no superfluous rules such that:*

1. *$\mathcal{R}$ is equivalent to $\mathcal{R}'$, AND*

2. *If R was shown invertible in $\mathcal{R}$ by one of our lemmata, and R is not superfluous, then R is invertible in $\mathcal{R}'$.*

Proof. The first part is a simple adaptation of the proof of Lemma 5 (p. 66).

For the second, suppose R was shown invertible in $\mathcal{R}$, and further that R was not superfluous. Take any premiss of R; it will be a premiss of every rule which shares the same principal formula. Since we have deleted rules going from $\mathcal{R}$ to $\mathcal{R}'$, then this premiss will still be a premiss of every rule which shares the same principal formula. Therefore, R is invertible in $\mathcal{R}'$. ⊣

Indeed, sometimes removing redundant rules will mean the remaining rules become invertible. Consider the calculus containing the two rules:

$$\frac{\Gamma \Rightarrow \phi, \Delta \quad \phi, \Gamma \Rightarrow \Delta}{\Gamma, !\phi \Rightarrow \Delta}\ L!_1 \qquad \frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, !\phi \Rightarrow \Delta}\ L!_2$$

If these were the two rules for ! on the left, then neither would be invertible. However, if we remove the superfluous $L!_1$, then $L!_2$ becomes invertible: $\Gamma \Rightarrow \Delta \oplus \phi$ is now a premiss of *every* rule with $!\phi$ principal on the left.

Thus, we have that removing superfluous rules will *not* harm invertibility; it can even aid invertibility. The removal of superfluous rules may not be depth-preserving. It will be depth-preserving if *Cut* is height preserving admissible.

6.5.2 Removal of Redundant Rules

A rule is redundant if it contains the premisses of another rule with the same conclusion. The removal of redundant rules gives an equivalent calculus with a smaller set of rules (see section 5.2). How will this affect invertibility of the rules?

Lemma 16 (Removing Redundant Rules) *Let $\mathcal{R}$ be a set of decomposable uniprincipal rules. Then, there exists a set of decomposable uniprincipal rules $\mathcal{R}'$ which contains no redundant rules such that:*

1. *$\mathcal{R}$ is (depth-preserving) equivalent to $\mathcal{R}'$, AND*

2. *If R was shown invertible in $\mathcal{R}$ by one of our lemmata, and R is not redundant, then R is invertible in $\mathcal{R}'$.*

Proof. Almost identical to the previous lemma. ⊣

Indeed, the removal of redundant rules can mean more rules become invertible. The same example can be used as before; $L!_2$ is redundant as well as superfluous.

Consider the example:

$$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_1 \qquad \frac{\Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_2 \qquad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_3$$

None of these rules are invertible. Removing the redundant rule $R\circ_3$ will not make the other two invertible. However, if we were to remove $R\circ_1$ and $R\circ_2$ then $R\circ_3$ would become invertible. Would this removal create an equivalent calculus? It would not. Consider the following derivation in the original calculus:

$$\frac{\overline{p_1 \Rightarrow p_1}}{p_1 \Rightarrow p_1 \circ p_2}\ r\circ_1$$

This conclusion is not derivable in the calculus with $R\circ_1$ and $R\circ_2$ removed. The closest one can achieve is:

$$\frac{\overline{p_1, p_2 \Rightarrow p_1} \quad \overline{p_1, p_2 \Rightarrow p_2}}{p_1, p_2 \Rightarrow p_1 \circ p_2}\ r\circ_3$$

Even appealing to *Cut* (assuming the system admits *Cut*), we would still require the sequent $p_1 \Rightarrow p_2$ to be derivable.

6.5.3 Transformation to Full Rules

Recall the example from section 5.2 when the rule for right disjunction from **G3cp** was changed to full rules:

$$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\ R\vee_1 \qquad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\ R\vee_2$$

$$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\ R\vee_3$$

None of these rules are invertible. Our lemmata say nothing about their invertibility. For instance, the left premiss of $R\vee_1$ is not a premiss of $R\vee_3$, so it will not satisfy the conditions of Lemma 8 (p. 82). So, we have replaced one invertible rule with three non-invertible rules.

Indeed, the process of creating a set of full rules will increase the number of rules with the same principal formula. Given our conditions, this will make the invertibility of such rules less likely.

What would be a good procedure is the *opposite*; squashing rules together. This will only be applicable if we have multisuccedent rules, and do not have context-dependent rules. The next section expands upon this idea.

6.6 Combinable Rules

Suppose a calculus had three rules:

$$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_1 \qquad \frac{\Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_2 \qquad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ_3$$

$R\circ_3$ is invertible[1] but it is not shown invertible by our lemmata. Suppose a derivation ended with $\Gamma \Rightarrow \Delta \oplus A \circ B$. It could have come from an instance of any of the three rules above. In the former two cases, we simply weaken with the appropriate formula in the premisses (B and A respectively), and in the latter case the result is immediate. $R\circ_1$ and $R\circ_2$ are neither superfluous or redundant, however it would appear they are not needed.

The following definition can be used for any type of calculus, but we specialise it to decomposable uniprincipal calculi composed only of normal rules:

Definition 22 (Combinable rules) *A pair of rules is* ***combinable*** *iff they have the same conclusion AND only differ in one premiss. In other words, combinable right rules are of the form (left rules are an obvious analogy):*

[1] If these are the only rules in which $\circ$ is the principal connective on the right.

$$\frac{\Gamma, \Gamma_1 \Rightarrow \Delta, \Delta_1 \quad \cdots \quad \Gamma, \Gamma_i \Rightarrow \Delta, \Delta_i \quad \cdots \quad \Gamma, \Gamma_n \Rightarrow \Delta, \Delta_n}{\Gamma \Rightarrow \Delta, \star_s(\vec{\Phi})}$$

$$\frac{\Gamma, \Gamma_1 \Rightarrow \Delta, \Delta_1 \quad \cdots \quad \Gamma, \Gamma_i' \Rightarrow \Delta, \Delta_i' \quad \cdots \quad \Gamma, \Gamma_n \Rightarrow \Delta, \Delta_n}{\Gamma \Rightarrow \Delta, \star_s(\vec{\Phi})}$$

*A **combined** rule is created by replacing a pair of combinable rules with a rule which has the same conclusion AND:*

- *Where the combinable rules have identical premisses, the new rule has this premiss, AND*
- *Where the combinable rules have different premisses, the new rule has a premiss whose active part is the two active parts added together.*

In other words, the combined rule from the pair of combinable rules above is:

$$\frac{\Gamma, \Gamma_1 \Rightarrow \Delta, \Delta_1 \quad \cdots \quad \Gamma, \Gamma_i, \Gamma_i' \Rightarrow \Delta, \Delta_i, \Delta_i' \quad \cdots \quad \Gamma, \Gamma_n \Rightarrow \Delta, \Delta_n}{\Gamma \Rightarrow \Delta, \star_s(\vec{\Phi})}$$

⊣

We can generalise the definition to allow IW rules. If at least one of the rules to be combined is an IW rule, the combined rule is an IW rule.

Obviously, this technique is only valid where we can weaken on both sides of the sequent arrow. Lemma 7 (p. 81), and its associated results in different chapters, provides the sufficient conditions for a calculus to height preserving admit *Weakening*. The following result is similar to those of section 5.2:

Lemma 17 (Removing Combinable Rules) *Let $\mathcal{R}$ be a multisuccedent calculus defined by a set of uniprincipal decomposable rules. Let $\mathcal{R}'$ be a calculus based on $\mathcal{R}$ where all pairs of combinable rules have been replaced a combined rule. If a sequent was derivable in $\mathcal{R}$, then it is derivable at the same height in $\mathcal{R}'$.*

Proof. Clearly we need only consider the cases where the final step in the derivation in $\mathcal{R}$ was an instance of a rule in a combinable pair. We show the case where the rule is a right rule, the left case is symmetric. Suppose wlog that the premiss which ensured the rule was combinable was the first one. Then, the derivation ends with:

$$\frac{\Gamma, \Gamma_1 \Rightarrow \Delta, \Delta_1 \quad \cdots \quad \Gamma, \Gamma_n \Rightarrow \Delta, \Delta_n}{\Gamma \Rightarrow \Delta, \star_s(\vec{B})}$$

Then, all of the premisses are derivable in $\mathcal{R}$. Suppose the other rule in the combinable pair has an active part with first premiss $\Gamma_1' \Rightarrow \Delta_1'$. *Weakening* is depth-preserving admissible by lemma 7, hence we weaken the first premiss with Γ_1' on the left and Δ_1' on the right. Then:

$$\dfrac{\dfrac{\Gamma, \Gamma_1 \Rightarrow \Delta, \Delta_1}{\Gamma, \Gamma_1, \Gamma_1' \Rightarrow \Delta, \Delta_1, \Delta_1'} \quad \cdots \quad \Gamma, \Gamma_n \Rightarrow \Delta, \Delta_n}{\Gamma \Rightarrow \Delta, \star_s(\vec{B})}$$

is a derivation in $\mathcal{R}'$. ⊣

Again, the above proof considers normal rules only. If IW rules are considered, the proof is similar.

We then have a lemma similar to lemmata 15 (p. 93) and 16 (p. 94):

Lemma 18 (Removing Combinable rules) *Let $\mathcal{R}$ be a multisuccedent calculus defined by a set of decomposable uniprincipal rules, and let $\mathcal{R}'$ be obtained from $\mathcal{R}$ by replacing all combinable pairs with combined rules. If $R \in \mathcal{R}$ was shown invertible by a lemma from section 6.2, and R is not part of a combinable pair, then R in invertible in $\mathcal{R}'$.*

Proof. Similar to lemma 15. ⊣

When the calculus $\mathcal{R}$ admits *Contraction* then removing combinable rules is conservative (in other words, no further sequents become provable). It is not known whether removing combinable rules from a calculus which does not admit *Contraction* is conservative.

6.6.1 Example

We have now three results which will aid in proof search. Consider the following calculus with five rules for one ternary connective $\circ$:

$$\dfrac{\Gamma \Rightarrow \Delta, \phi_1}{\Gamma \Rightarrow \Delta, \circ(\phi_1, \phi_2, \phi_3)}\ R\circ_1 \qquad \dfrac{\Gamma \Rightarrow \Delta, \phi_2}{\Gamma \Rightarrow \Delta, \circ(\phi_1, \phi_2, \phi_3)}\ R\circ_2$$

$$\dfrac{\Gamma \Rightarrow \Delta, \phi_1, \phi_2 \quad \Gamma \Rightarrow \Delta, \phi_3}{\Gamma \Rightarrow \Delta, \circ(\phi_1, \phi_2, \phi_3)}\ R\circ_3$$

$$\dfrac{\Gamma \Rightarrow \Delta, \phi_1 \quad \Gamma, \phi_1 \Rightarrow \Delta \quad \Gamma \Rightarrow \Delta, \phi_3}{\Gamma, \circ(\phi_1, \phi_2, \phi_3) \Rightarrow \Delta}\ L\circ_1 \qquad \dfrac{\Gamma, \phi_1, \phi_2 \Rightarrow \Delta}{\Gamma, \circ(\phi_1, \phi_2, \phi_3) \Rightarrow \Delta}\ L\circ_2$$

Which rules are invertible? Only $L\circ_2$, and not by the methods of section 6.2. The left premiss of $R\circ_3$ is derivable at a height not greater than that of the conclusion $R\circ_3$.

Firstly, $R\circ_1$ and $R\circ_2$ form a combinable pair. Replace them with the combined rule $R\circ'$:

$$\dfrac{\Gamma \Rightarrow \Delta, \phi_1, \phi_2}{\Gamma \Rightarrow \Delta, \circ(\phi_1, \phi_2, \phi_3)}\ R\circ'$$

Now $R\circ_3$ is redundant, so we can remove it. This leaves the rule $R\circ'$, which is invertible by lemma 8.

The calculus consisting of $R\circ'$, $L\circ_1$ and $L\circ_2$ admits multi-cut, or *Cut* for structures. In other words, the following rule is admissible:

$$\frac{\Gamma \Rightarrow \Delta, \Theta \quad \Gamma, \Theta \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ Cut_m$$

where Θ is some multiset of formulae. From this, one can deduce that *Cut* (for single formulae) is admissible (see, for instance, [Shoesmith and Smiley, 2008]). Then, we can remove the superfluous rule $L\circ_1$ to leave a calculus with just $R\circ'$ and $L\circ_2$ as the only rules. Using lemmata 8 (p. 82) and 9 (p. 85) the calculus is invertible.

6.7 More complex propositional calculi

As noted in section 5.5, the rule $L0\supset$ from **G4ip** precludes that calculus from the analysis in the previous sections. Here, we relax the condition that the active part of rules may only have one compound formula in their conclusions. Now, however, *every* metaformula which appears in the conclusion of the active part is principal for the rule. As an example, both P and $P\supset\phi$ are principal for $L0\supset$.

We can then reconstruct similar lemmata to those of the previous sections, however we must be careful with respect to atoms and $\bot$. We have the following:

Lemma 19 (Multiprincipal Rules) *Let $\mathcal{R}$ be a set of decomposable rules. Then, the rule:*

$$\frac{\Gamma, \Gamma'' \Rightarrow \Delta, \Delta''}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

is height preserving admissible if:

1. *For every $\phi \in \Gamma'', \Delta''$, we have $\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of every rule with ϕ principal.*
2. *Every atom variable in Γ'' (Δ'') occurs in Γ' (Δ').*
3. *If $\bot \in \Gamma''$, then $\bot \in \Gamma'$.*

Proof. Let $\Gamma + \Gamma'' \Rightarrow \Delta + \Delta''$ be an instantiation of the premiss. We proceed by induction on the height, n, of the derivation of the premiss. If $n = 0$, then the premiss was an axiom. Conditions 2 and 3 guarantee the conclusion is also an axiom.

If $n > 0$, let r be the last inference used in the derivation. Then there are four cases:

1. r is an instance of a normal rule, and there is some $A \in \Gamma'', \Delta''$ which is principal for r.

2. r is an instance of an IW rule, and there is some $A \in \Gamma'', \Delta''$ which is principal for r.

3. r is an instance of a normal rule, and there is no $A \in \Gamma'', \Delta''$ which is principal for r.

4. r is an instance of an IW rule, and there is no $A \in \Gamma'', \Delta''$ which is principal for r.

Case 1. $\Gamma' \Rightarrow \Delta'$ is the active part of a premiss of r. If the active part of r had conclusion $\Gamma'' \Rightarrow \Delta''$, then the result is immediate. There may be other inferences with active parts of the form:

$$\frac{\cdots \quad \Gamma' \Rightarrow \Delta' \quad \cdots}{\Gamma'', \Gamma''' \Rightarrow \Delta'', \Delta'''}$$

In these cases, we have $\Gamma''' \subseteq \Gamma$ and $\Delta''' \subseteq \Delta$. So, there exist $\bar{\Gamma}$ and $\bar{\Delta}$ such that $\Gamma = \Gamma''' + \bar{\Gamma}$ and $\Delta = \Delta''' + \bar{\Delta}$. We can then rewrite r as:

$$\frac{\cdots \quad \bar{\Gamma}, \Gamma' \Rightarrow \bar{\Delta}, \Delta' \quad \cdots}{\bar{\Gamma}, \Gamma'', \Gamma''' \Rightarrow \bar{\Delta}, \Delta'', \Delta'''}$$

Weakening with Γ''' and Δ''' gives the required result; the premisses are derivable at height $n - 1$.

Case 2. This case is similar to the equivalent case in the proofs of lemmata 8 (p. 82), 9 (p. 85) and case 1.

Case 3. Since *no* formula from Γ'', Δ'' is principal for r, all such formulae must be in the context of r. Then, we reason in the same fashion as in the proofs of lemmata 8 and 9.

Case 4. We know *no* formulae from Γ'', Δ'' are principal for r. We have r will be of the form:

$$\frac{\Gamma_1, \Gamma_1'' \Rightarrow \Delta_1, \Delta_1'' \quad \cdots \quad \Gamma_1, \Gamma_n'' \Rightarrow \Delta_1, \Delta_n''}{\Gamma_1, \Gamma_2, \Gamma_\star'' \Rightarrow \Delta_1, \Delta_2, \Delta_\star''}$$

Either Γ'' is in Γ_1, or Γ_2, or partially contained in both. We reason by cases. Either, $\Gamma'' \nsubseteq \Gamma_2$, and $\Delta'' \nsubseteq \Delta_2$, or at least some part of Γ'' or Δ'' is in Γ_1 or Δ_1, respectively.

In the former case, the situation is similar to that in lemma 8 and lemma 9; we use a new inference which is the same as r except that the implicit weakening part of the inference contains Γ' and Δ', instead of Γ'' and Δ''.

In the other case, we perform two steps. Firstly, we weaken every premiss of r so that the context of each premiss contains Γ'' and Δ''. This will be the same weakening for every premiss, because every premiss contains Γ_1 and Δ_1, and hence the same elements of Γ'' and Δ''. Then, we apply the induction hypothesis to each of these new premisses, so that we remove Γ'' and Δ'' and replace them with Γ' and Δ' in the context of each premiss.

Then, we remove from Γ_2 and Δ_2 any formulae which occur in Γ'' and Δ'', respectively, and we will use this as our new implicit weakening. Applying the new inference with this

information will yield $\Gamma + \Gamma' \Rightarrow \Delta + \Delta'$ as derivable at height n, and this completes the case, and the proof. $\dashv$

In particular, the results from the previous sections are now specialisations of this result.

6.7.1 Examples

When restricted to single formulae on the right in an analogous way to section 6.3, **G4ip** is a calculus in which all rules have the form given above. The rule which did not allow **G4ip** to be classified as a uniprincipal calculus was $L0\supset$, shown above. Now we can see, however, that it is indeed invertible; the atom variable P is retained in the premiss, and this rule is the only rule where $P \supset \phi$ will be principal on the left. In what follows, the notation $\wr A_1, \ldots, A_n \wr$ is used to denote the multiset containing the formulae $A_1, \ldots, A_n$.

Consider the calculus **G3-LC** for Gödel-Dummett logic from [Sonobe, 1975]. Axioms are given as normal, the only logical connective is $\supset$ and the rules for it are:

$$\frac{\Gamma, \phi \supset \psi \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \supset \psi \Rightarrow \Delta}\ L\supset$$

$$\frac{\{\Gamma, \phi_i \Rightarrow \psi_i, \Delta^i\}}{\Gamma \Rightarrow \Delta}\ R\supset$$

where $i = 1, \ldots, m$ and:

$$\Delta^i = \wr \phi_1 \supset \psi_1, \ldots, \phi_{i-1} \supset \psi_{i-1}, \phi_{i+1} \supset \psi_{i+1}, \ldots, \phi_m \supset \psi_m \wr$$

Δ contains this multiset and may contain other formulae as well. When $m = 2$, for instance, the rule will be:

$$\frac{\Gamma, \phi_1 \Rightarrow \psi_1, \phi_2 \supset \psi_2 \quad \Gamma, \phi_2 \Rightarrow \psi_2, \phi_1 \supset \psi_1}{\Gamma \Rightarrow \phi_1 \supset \psi_1, \phi_2 \supset \psi_2, \Delta'}$$

For a given m, this rule is an IW multiprincipal rule. In the case where Δ' is empty, then the rule is invertible. In the case where Δ' is non-empty, then we can use lemma 19 to show that each premiss, weakened on the right with Δ', is derivable without an increase in height.

As a more involved example, consider the calculus with the two binary connectives $\circ$ and $\star$, and the six rules:

$$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \circ \psi \Rightarrow \Delta}\ L\circ \qquad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \circ \psi, \Delta}\ R\circ$$

$$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma, \phi \star \psi \Rightarrow \Delta}\ L\star \qquad \frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \star \psi, \Delta}\ R\star$$

$$\frac{\Gamma, \phi, \psi \Rightarrow \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma, \phi \circ \psi \Rightarrow \phi \star \psi, \Delta}\ L_\circ R_\star$$

$$\frac{\Gamma, \psi \Rightarrow \phi, \Delta}{\Gamma, \phi \star \psi \Rightarrow \phi \circ \psi, \Delta}\ L_\star R_\circ$$

The following rules are height preserving admissible:

$$\frac{\Gamma, \phi \circ \psi \Rightarrow \Delta}{\Gamma, \phi, \psi \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \phi \star \psi, \Delta}{\Gamma \Rightarrow \psi, \Delta}$$

Take the left-hand rule. $\phi \circ \psi$ will be left principal in rules $L\circ$ and $L_\circ R_\star$. For both of these rules $\phi \oplus \psi \Rightarrow \emptyset$ is the active part of a premiss. If the last inference was based on $L_\circ R_\star$, then $A \star B$ (the instantiation of $\phi \star \psi$) would be part of Δ, and so would disappear from the premiss. We would thus weaken with $A \star B$ (as in the latter part of case 1 of the proof, above).

The rule:

$$\frac{\Gamma \Rightarrow \phi \star \psi, \Delta}{\Gamma, \phi \Rightarrow \Delta}$$

cannot be shown admissible by use of lemma 19, for $\phi \Rightarrow \emptyset$ is not the active part of a premiss of $L_\circ R_\star$. Neither of the multiprincipal rules are invertible.

6.8 Invertible Sets of Rules

Using the results from the previous sections, we can analyse individual rules in certain types of calculi. The result we give in this section is specific to decomposable uniprincipal multisuccedent calculi, but it can be extended where appropriate.

Recall we said a rule is height preserving invertible (definition 17, page 77) if every premiss of the rule is derivable at a height not greater than the height of a derivation of the conclusion of the rule. We check invertibility by searching among the premisses of rules with the same principal formula. Clearly, if no two rules have the same principal formula, then this task becomes much more straightforward. This motivates the following definition:

Definition 23 (Unique Conclusion Property) *A set of decomposable rules $\mathcal{R}$ has the* ***unique conclusion property*** *iff for all S and T in $\mathcal{R}$, if the conclusion of the active part of S is the same as the conclusion of the active part of T, then $S = T$.* ⊣

In [Curry, 1963], it was proved (for multisuccedent propositional calculi) that if a set of rules had the unique conclusion property, then every rule would be invertible. This becomes an obvious consequence of the lemmata in section 6.2:

Lemma 20 (Multisuccedent Invertible Rule Sets) *Let $\mathcal{R}$ be a set of decomposable uniprincipal rules defining a sequent calculus. If $\mathcal{R}$ has the unique conclusion property and:*

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_n \Rightarrow \Delta_n}{\Gamma \Rightarrow \Delta}$$

is an instance of a rule of $\mathcal{R}$ and $\Gamma \Rightarrow \Delta$ is derivable with respect to $\mathcal{R}$ at height m, then $\Gamma_i \Rightarrow \Delta_i$ is derivable at height at most m for $i = 1, \ldots, n$.

Proof. Let $1 \leq i \leq n$ and m be given. Let R be the rule of which $\Gamma_i \Rightarrow \Delta_i$ is an instantiated premiss. R will have some principal formula, say ϕ. From the unique conclusion property, $\Gamma_i \Rightarrow \Delta_i$ is a premiss of *every* rule with ϕ principal. Depending on whether ϕ is left or right principal, we therefore apply one of the lemmata from section 6.2, and we have that $\Gamma_i \Rightarrow \Delta_i$ is derivable at a height not greater than m, as required. ⊣

We can thus very quickly check whether a set of rules is invertible. For instance, **G3cp** is an invertible set of rules: for each connective, there is only one left and one right rule.

The reason this result is so clean for multisuccedent uniprincipal propositional calculi is that the condition set down in the lemmata of section 6.2 was simple. We can extend the result to the other types of calculus considered, by changing the conditions on the rule set. However, the restrictions on first-order and modal calculi would be so restrictive they would be expressively weak. Thus, we only show the single succedent restrictions.

6.8.1 Single Succedent calculi

It is obvious that lemma[2] 20 will not be true for single succedent calculi. The implicational fragment of intuitionistic logic provides a counter-example. The rules consist only of:

$$\frac{\Gamma, \phi \supset \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \sigma}{\Gamma, \phi \supset \psi \Rightarrow \sigma}\ L\supset \qquad \frac{\Gamma, \phi \Rightarrow \psi}{\Gamma \Rightarrow \phi \supset \psi}\ R\supset$$

Clearly this calculus has the unique conclusion property, but it is not invertible. For, consider the derivation:

$$\frac{}{P, A \supset B \Rightarrow P}$$

[2] If naïvely restricted to single succedents.

If the left rule were invertible, then the sequent $P \oplus A \supset B \Rightarrow A$ would be derivable at height 0. However, it is not derivable unless $A = P$.

We thus need to generalise the conditions of lemma 11 (p. 87) so that it can be applied across a rule set.

Lemma 21 (Single Succedent Invertible Rule Sets) *Let $\mathcal{R}$ be a set of single succedent decomposable uniprincipal rules defining a sequent calculus. If $\mathcal{R}$ has the unique conclusion property* ***and*** *$\mathcal{R}$ contains no left rules which have a premiss with an active succedent, then $\mathcal{R}$ is invertible.*

Proof. Any rule in such a calculus will be of the necessary form for lemma 11. ⊣

In particular, rules such as $L \supset$ are not allowed to be in such a calculus. Only the conjunctive fragment of **G3ip** is invertible; the disjunctive fragment does not have the unique conclusion property.

As an example, consider the calculus with one ternary connective $\circ$, and two rules:

$$\frac{\Gamma \Rightarrow \phi_1 \quad \Gamma, \phi_2 \Rightarrow \phi_3}{\Gamma \Rightarrow \circ(\phi_1, \phi_2, \phi_3)}\ R\circ \qquad \frac{\Gamma, \phi_1 \Rightarrow \sigma \quad \Gamma, \phi_2, \phi_3 \Rightarrow \sigma}{\Gamma, \circ(\phi_1, \phi_2, \phi_3) \Rightarrow \sigma}\ L\circ$$

This is an invertible set of rules. Using the result from section 5.6, it will be possible to permute non-principal instances of *Cut* into the premisses of the left rule. Thus, we only need to look at principal instances of *Cut*. In this case, *Cut* will be admissible, since we can cut ϕ_1 using the left premisses of $R\circ$ and $L\circ$:

$$\frac{\Gamma \Rightarrow \phi_1 \quad \Gamma, \phi_1 \Rightarrow \sigma}{\Gamma \Rightarrow \sigma}\ IH$$

6.9 Conclusions

The results of this chapter were all novel. The new framework developed in the previous chapter was the basis for these results.

We have given, for a wide-range of calculi, sufficient conditions for the invertibility of rules. For a finite set of rules, the conditions are easily verified. Moreover, whilst the particulars of the conditions vary throughout the calculi, the general idea is the same for all: a rule is invertible if its premisses occur as premisses of every rule with the same principal formula.

The results of section 6.5 and section 6.6 are useful when designing calculi. Proof search will succeed most effectively when there is a small set of rules; the results given within those sections show how to prune a set of rules, whilst not losing any expressive power.

Of course, one of the stated aims of this thesis was to explore how structural proof theory could be adequately performed in *Isabelle*. Should it be possible to formalise the results of this chapter in a straightforward manner, then one would take a large step towards removing

the inefficiency which was noted in chapter 4. It is this problem which will occupy us in the next chapter.

Chapter 7

A Formalisation

7.1 Introduction

In this chapter, we give an overview of the results of the previous sections formalised in *Isabelle*. Since the results of those sections were novel, the formalisations contained in this chapter are likewise novel. The full proof scripts are online [Chapman, 2009]: we only show the pertinent details here.

The framework is outlined in section 7.2, along with a discussion of why the methods of this chapter and section 5.5 coincide. As with the previous chapter, the formalisation is given primarily for the uniprincipal multisuccedent calculi. We will use, as our running example throughout, the calculus **G3cp**. In section 7.4, we look at the formalisation of single-succedent calculi; in section 7.5 the results for modal logic are examined. We return to uniprincipal multisuccedent calculi in section 7.6 to look at manipulating rule sets.

7.2 Formalising the Framework

In section 5.5, we defined calculi by giving the decomposable rules and extracting active parts. It is also possible to give the active parts themselves, and define the calculus as the extension of those active parts. By extension, we mean adding of formulae in some fashion. We will consider two methods of extension: normal and IW extensions. In the former, we add the multisets Γ and Δ to the antecedent and succedent, respectively, of *every* premiss and the conclusion. In the latter, we also add the extra multisets Γ' and Δ' to the antecedent and the succedent of the conclusion. Will these two approaches coincide?

Suppose we had a decomposable calculus $\mathcal{R}$, then let $\hat{\mathcal{R}}$ be the active parts of the rules of $\mathcal{R}$. Similarly, if $\mathcal{A}$ is a set of active parts, let $\mathcal{A}^\star$ be the extension of those active parts with either normal or IW passive parts. Showing the two approaches coincide equates to showing the following equivalences:

$$(\hat{\mathcal{R}})^\star = \mathcal{R} \qquad \text{where } \mathcal{R} \text{ is a decomposable calculus}$$

$$\widehat{(\mathcal{A}^\star)} = \mathcal{A} \qquad \text{where } \mathcal{A} \text{ is a set of active parts}$$

When $\mathcal{A} = \hat{\mathcal{R}}$, it does not matter which approach one uses.

Lemma 22 (Extracting then Extending Active Parts) *Let $\mathcal{R}$ be a decomposable calculus. If $\mathcal{R} = \mathcal{R}_n \cup \mathcal{R}_{IW}$, where $\mathcal{R}_n$ ($\mathcal{R}_{IW}$) contains normal (IW) rules only, then:*

$$(\hat{\mathcal{R}})^\star = \mathcal{R}$$

Proof. Let $R \in \mathcal{R}$ be given. Since $\mathcal{R}$ is a decomposable calculus, R will be composed of an active and passive part (of the appropriate kind, depending on whether $R \in \mathcal{R}_n$ or $R \in \mathcal{R}_{IW}$). This active part will be, by definition, in $\hat{\mathcal{R}}$. So, the extension of this active part will be in $(\hat{\mathcal{R}})^\star$, thus $R \in (\hat{\mathcal{R}})^\star$. Rewinding this chain gives that, if $R \in (\hat{\mathcal{R}})^\star$ then $R \in \mathcal{R}$. ⊣

Lemma 23 (Extending then Extracting Active Parts) *Let $\mathcal{A}$ be a set of active parts. If $\mathcal{A} = \mathcal{A}_n \cup \mathcal{A}_{IW}$, where $\mathcal{A}_n$ ($\mathcal{A}_{IW}$) contains active parts which are to be extended in a normal (IW) way, then:*

$$\widehat{(\mathcal{A}^\star)} = \mathcal{A}$$

Proof. Similar to that of lemma 22 (p. 106). ⊣

In both proofs, that rules are separated by the type of extension they require is necessary. For, suppose we did not have this requirement. We may extract an active part from a normal rule, but then extend in an IW fashion. We would have thus not regained the original calculus (unless, of course, the rule appeared in both normal and IW flavours).

7.2.1 Formulae and Sequents

A *formula* is either a propositional variable, the constant $\bot$, or a connective applied to a list of formulae. We thus have a type variable indexing formulae, where the type variable will be a set of connectives. In the usual way, we index propositional variables by use of natural numbers. There are no arity constraints for the connectives and the lists at this stage. So, formulae are given by the datatype:

datatype *'a form = At nat*
| Compound 'a ('a form list)
| ff

For **G3cp**, we define the datatype *Gp*, and give the following abbreviations:

datatype *Gp = con | dis | imp*

types *Gp-form* = *Gp form*

abbreviation *con-form* (**infixl** $\wedge*$ *80*) **where**
$p \wedge* q \equiv$ *Compound con* [*p*,*q*]

abbreviation *dis-form* (**infixl** $\vee*$ *80*) **where**
$p \vee* q \equiv$ *Compound dis* [*p*,*q*]

abbreviation *imp-form* (**infixl** $\supset$ *80*) **where**
$p \supset q \equiv$ *Compound imp* [*p*,*q*]

Giving the concrete example imposes arity constraints: it is impossible now for a conjunction to take a list with three formulae as its argument.

A *sequent* is a pair of multisets of formulae. Sequents are indexed by the connectives used to index the formulae. Recall that to add a single formula to a multiset of formulae, we use the symbol $\oplus$, whereas to join two multisets, we use the symbol $+$.

In section 5.5 we introduced metaformulae as well as formulae. In *Isabelle*, the "'a form" will play the role of metaformulae. There is a difference between what *Isabelle* calls *schematic variables* and the concrete members of the datatypes. When we prove a lemma, or write an inductive definition, it is then available to use as a simplification, or rewrite, rule. It is stated in terms of schematic variables, which are prefixed by ?, which we can then instantiate.

7.2.2 Rules and Rule Sets

A *rule* is a list of sequents (called the premisses) paired with a sequent (called the conclusion). The two *rule sets* used for uniprincipal multisuccedent calculi are the axioms, and the uniprincipal rules. Both are defined as inductive sets. There are two clauses for axioms, corresponding to $L\bot$ and normal axioms:

inductive-set *Ax* **where**
id: ([], $\wr$ *At i* $\int \Rightarrow*$ $\wr$ *At i* $\int$) $\in$ *Ax*
| *Lbot*: ([], $\wr$ *ff* $\int \Rightarrow* \emptyset$) $\in$ *Ax*

The set of uniprincipal rules, on the other hand, must not have empty premisses[1], and must have a single, compound formula in its conclusion. The function `mset` takes a sequent, and returns the multiset obtained by adding the antecedent and the succedent together:

inductive-set *upRules* **where**
I: $[\![$ *mset c* $\equiv \wr$ *Compound R Fs* $\int$; *ps* $\neq$ [] $]\!] \Longrightarrow$ (*ps*,*c*) $\in$ *upRules*

For **G3cp**, we have the following six rules (each A and B below are *Isabelle* schematic

[1] It is possible to remove this constraint to allow zero premiss rules. However, this restriction makes it straightforward to distinguish between the set of uniprincipal rules and axioms.

variables, which stand for metaformulae), which we then show are a subset of the set of uniprincipal rules:

inductive-set *g3cp*
where
  conL: $([⟅ A ⟆ + ⟅ B ⟆ \Rightarrow* \emptyset], ⟅ A \wedge* B ⟆ \Rightarrow* \emptyset) \in g3cp$
| *conR*: $([\emptyset \Rightarrow* ⟅ A ⟆, \emptyset \Rightarrow* ⟅ B ⟆], \emptyset \Rightarrow* ⟅ A \wedge* B ⟆) \in g3cp$
| *disL*: $([⟅ A ⟆ \Rightarrow* \emptyset, ⟅ B ⟆ \Rightarrow* \emptyset], ⟅ A \vee* B⟆ \Rightarrow* \emptyset) \in g3cp$
| *disR*: $([\emptyset \Rightarrow* ⟅ A ⟆ + ⟅ B ⟆], \emptyset \Rightarrow* ⟅ A \vee* B ⟆) \in g3cp$
| *impL*: $([\emptyset \Rightarrow* ⟅ A ⟆, ⟅ B ⟆ \Rightarrow* \emptyset], ⟅ A \supset B ⟆ \Rightarrow* \emptyset) \in g3cp$
| *impR*: $([⟅ A ⟆ \Rightarrow* ⟅ B ⟆], \emptyset \Rightarrow* ⟅ A \supset B ⟆) \in g3cp$

lemma *g3cp-upRules*:
shows $g3cp \subseteq upRules$
proof−
 {
  fix *ps c*
  assume $(ps,c) \in g3cp$
  then have $(ps,c) \in upRules$ **by** (*induct*) *auto*
 }
thus $g3cp \subseteq upRules$ **by** *auto*
qed

We have thus given the active parts of the **G3cp** calculus. We now need to extend these active parts with passive parts.

Given a sequent C, we extend it with another sequent S by adding the two antecedents and the two succedents. To extend a normal active part[2] (Ps, C) with a sequent S, we extend every $P \in Ps$ and C with S:

defs *extend-def* : *extend forms seq* ≡
  (*antec forms* + *antec seq*) ⇒* (*succ forms* + *succ seq*)
defs *extendRule-def* : *extendRule forms R* ≡
  (*map* (*extend forms*) (*fst R*), *extend forms* (*snd R*))

Given a rule set $\mathcal{R}$, the *extension* of $\mathcal{R}$, called $\mathcal{R}^\star$, is then defined as another inductive set:

inductive-set *extRules* :: *'a rule set* ⇒ *'a rule set* (-*)
 for *R* :: *'a rule set*
 where *I*: $r \in R \Longrightarrow$ *extendRule seq r* $\in R*$

G3cp has the unique conclusion property (see definition 23). This is easily formalised:

defs *uniqueConclusion-def* : *uniqueConclusion R* ≡ ∀ *r1* ∈ *R*. ∀ *r2* ∈ *R*.
  (*snd r1* = *snd r2*) ⟶ (*r1* = *r2*)

lemma *g3cp-uc*:

[2] In this section, we only use normal rules. In the section on modal logics, we will use IW rules.

shows *uniqueConclusion g3cp*
apply (*auto simp add*:*uniqueConclusion-def Ball-def*)
apply (*rule g3cp.cases*) **apply** *auto* **by** (*rotate-tac 1*,*rule g3cp.cases*,*auto*)+

7.2.3 Principal Rules and Derivations

A formula A is *left principal* for an active part R iff the conclusion of R is of the form $A \Rightarrow \emptyset$. The definition of *right principal* is then obvious. We have an inductive predicate to check these things:

inductive *rightPrincipal* :: *'a rule* ⇒ *'a form* ⇒ *bool*
where
up: $C = (\emptyset \Rightarrow\!* \lbag Compound\ F\ Fs \rbag) \Longrightarrow$
rightPrincipal (*Ps*,*C*) (*Compound F Fs*)

As an example, we show that if $A \wedge B$ is principal for an active part in **G3cp**, then $\emptyset \Rightarrow A$ is a premiss of that active part:

lemma *principal-means-premiss*:
assumes *a*: *rightPrincipal r* (*A* ∧* *B*)
and *b*: *r* ∈ *g3cp*
shows (∅ ⇒* ⦇ *A* ⦈) ∈ *set* (*fst r*)
proof−
from *a* **and** *b* **obtain** *Ps* **where** *req*: *r* = (*Ps*, ∅ ⇒* ⦇ *A*∧**B* ⦈)
by (*cases r*) *auto*
with *b* **have** *Ps* = [∅ ⇒* ⦇ *A* ⦈, ∅ ⇒* ⦇ *B* ⦈]
apply (*cases r*) **by** (*rule g3cp.cases*) *auto*
with *req* **show** (∅ ⇒* ⦇ *A* ⦈) ∈ *set* (*fst r*) **by** *auto*
qed

A sequent is *derivable* at height 0 if it is the conclusion of a rule with no premisses. If a rule has m premisses, and the maximum height of the derivation of any of the premisses is n, then the conclusion will be derivable at height $n+1$. We encode this as pairs of sequents and natural numbers, and call such a pair a *deriv*. A sequent S is derivable at a height n in a rule system $\mathcal{R}$ iff (S, n) belongs to the inductive set `derivable` $\mathcal{R}$:

inductive-set *derivable* :: *'a rule set* ⇒ *'a deriv set*
for *R* :: *'a rule set*
where
base: ⟦([],*C*) ∈ *R*⟧ ⟹ (*C*,*0*) ∈ *derivable R*
| *step*: ⟦ *r* ∈ *R* ; (*fst r*)≠[] ; ∀ *p* ∈ *set* (*fst r*). ∃ *n* ≤ *m*. (*p*,*n*) ∈ *derivable R* ⟧
⟹ (*snd r*,*m* + *1*) ∈ *derivable R*

In some instances, we do not care about the height of a derivation, rather that the root

is derivable. For this, we have the additional definition of `derivable'`, which is a set of sequents:

inductive-set *derivable'* :: *'a rule set* ⇒ *'a sequent set*
  for *R* :: *'a rule set*
  where
   base: ⟦ ([],*C*) ∈ *R* ⟧ ⟹ *C* ∈ *derivable' R*
| *step*: ⟦ *r* ∈ *R* ; (*fst r*) ≠ [] ; ∀ *p* ∈ *set* (*fst r*). *p* ∈ *derivable' R* ⟧
                ⟹ (*snd r*) ∈ *derivable' R*

It is desirable to switch between the two notions. Shifting from derivable at a height to derivable is simple: we delete the information about height. The converse is more complicated and involves an induction on the length of the premiss list:

lemma *deriv-to-deriv*:
assumes (*C*,*n*) ∈ *derivable R*
shows *C* ∈ *derivable' R*
using *assms* **by** (*induct*) *auto*

lemma *deriv-to-deriv2*:
assumes *C* ∈ *derivable' R*
shows ∃ *n*. (*C*,*n*) ∈ *derivable R*
using *assms*
 proof (*induct*)
 case (*base C*)
 then have (*C*,*0*) ∈ *derivable R* **by** *auto*
 then show *?case* **by** *blast*
next
 case (*step r*)
 then obtain *ps c* **where** *r* = (*ps*,*c*) **and** *ps* ≠ [] **by** (*cases r*) *auto*
 then have *aa*: ∀ *p* ∈ *set ps*. ∃ *n*. (*p*,*n*) ∈ *derivable R* **by** *auto*
 then have ∃ *m*. ∀ *p* ∈ *set ps*. ∃ *n*≤*m*. (*p*,*n*) ∈ *derivable R*
    proof (*induct ps*) — induction on the list
    case *Nil*
    then show *?case* **by** *auto*
 next
    case (*Cons a as*)
    then have ∃ *m*. ∀ *p* ∈ *set as*. ∃ *n*≤*m*. (*p*,*n*) ∈ *derivable R* **by** *auto*
    then obtain *m* **where** ∀ *p* ∈ *set as*. ∃ *n*≤*m*. (*p*,*n*) ∈ *derivable R* **by** *auto*
    moreover from ‹∀ *p* ∈ *set* (*a* # *as*). ∃ *n*. (*p*,*n*) ∈ *derivable R*› **have**
          ∃ *n*. (*a*,*n*) ∈ *derivable R* **by** *auto*
    then obtain *m'* **where** (*a*,*m'*) ∈ *derivable R* **by** *blast*
    ultimately have ∀ *p* ∈ *set* (*a* # *as*). ∃ *n*≤(*max m m'*). (*p*,*n*) ∈ *derivable R*
by *auto* — max returns the maximum of two integers

```
    then show ?case by blast
    qed
  then obtain m where ∀ p ∈ set ps. ∃ n≤m. (p,n) ∈ derivable R by blast
  with ⟨r = (ps,c)⟩ and ⟨r ∈ R⟩ have (c,m+1) ∈ derivable R using ⟨ps ≠ []⟩ and
    derivable.step[where r=(ps,c) and R=R and m=m] by auto
  then show ?case using ⟨r = (ps,c)⟩ by auto
qed
```

7.3 Formalising the Results

We now have all of the foundations required to formalise the results of section 6.2 for an arbitrary language. The example which runs throughout this section shows the general results applied to the encoding of the **G3cp** calculus. A variety of "helper" lemmata are used in the proofs, but are not shown here. Full details, including all the proofs and helper lemmata, can be found at [Chapman, 2009]. The proof tactics themselves are hidden in the following proof, except where they are interesting. Indeed, only the interesting parts of the proof are shown at all. We formalise lemma 8 (p. 82), except that we have the restriction that all rules are normal. The proof is interspersed with comments.

```
lemma rightInvertible:
fixes Γ Δ :: 'a form multiset
assumes rules: R' ⊆ upRules ∧ R = Ax ∪ R'
  and   a: (Γ ⇒* Δ ⊕ Compound F Fs,n) ∈ derivable R*
  and   b: ∀ r' ∈ R. rightPrincipal r' (Compound F Fs) ⟶
           (Γ' ⇒* Δ') ∈ set (fst r')
shows ∃ m≤n. (Γ +Γ' ⇒* Δ + Δ',m) ∈ derivable R*
using assms
```

The height of derivations is decided by the length of the longest branch. Thus, we need to use strong induction: i.e. $\forall m \leq n$. If $P(m)$ then $P(n+1)$.

```
proof (induct n arbitrary:Γ Δ rule:nat-less-induct)
 case (1 n Γ Δ)
 then have IH:∀ m<n. ∀Γ Δ. ( Γ ⇒* Δ ⊕ Compound F Fs, m) ∈ derivable R* ⟶
                (∀ r' ∈ R. rightPrincipal r' (Compound F Fs) ⟶
                ( Γ' ⇒* Δ') ∈ set (fst r')) ⟶
                (∃ m'≤m. ( Γ + Γ' ⇒* Δ + Δ', m') ∈ derivable R*)
    and a': (Γ ⇒* Δ ⊕ Compound F Fs,n) ∈ derivable R*
    and b': ∀ r' ∈ R. rightPrincipal r' (Compound F Fs) ⟶
                (Γ' ⇒* Δ') ∈ set (fst r')
      by auto
 show ?case
 proof (cases n)   — Case analysis on n
```

case *0*
then obtain r S **where** *extendRule* S r = ([],Γ $\Rightarrow*$ Δ $\oplus$ *Compound F Fs*)
and $r \in Ax \vee r \in R'$ **by** *auto* — At height 0, the premisses are empty
moreover
{assume $r \in Ax$
then obtain i **where** ([], ⦇ *At i* ⦈ $\Rightarrow*$ ⦇ *At i* ⦈) = r $\vee$
r = ([], ⦇ *ff* ⦈ $\Rightarrow*$ $\emptyset$)
using *characteriseAx*[**where** r=r] **by** *auto*
moreover — Case split on the kind of axiom used
{assume r = ([], ⦇ *At i* ⦈ $\Rightarrow*$ ⦇ *At i* ⦈)
then have *At i* :# Γ $\wedge$ *At i* :# Δ **by** *auto*
then have *At i* :# Γ + Γ' $\wedge$ *At i* :# Δ + Δ' **by** *auto*
then have (Γ + Γ' $\Rightarrow*$ Δ + Δ',*0*) $\in$ *derivable* $R*$ **using** *rules* **by** *auto*
}
moreover
{assume r = ([],⦇*ff*⦈ $\Rightarrow*$ $\emptyset$)
then have *ff* :# Γ **by** *auto*
then have *ff* :# Γ + Γ' **by** *auto*
then have (Γ + Γ' $\Rightarrow*$ Δ + Δ',*0*) $\in$ *derivable* $R*$ **using** *rules* **by** *auto*
}
ultimately have (Γ + Γ' $\Rightarrow*$ Δ + Δ',*0*) $\in$ *derivable* $R*$ **by** *blast*
}
moreover
{assume $r \in R'$ — This leads to a contradiction
then obtain *Ps C* **where** *Ps* $\neq$ [] **and** r = (*Ps*,*C*) **by** *auto*
moreover obtain S **where** r = ([],S) **by** *blast* — Contradiction
ultimately have (Γ + Γ' $\Rightarrow*$ Δ + Δ',*0*) $\in$ *derivable* $R*$ **using** *rules* **by** *simp*
}
ultimately show $\exists$ $m{\leq}n$. (Γ + Γ' $\Rightarrow*$ Δ + Δ',m) $\in$ *derivable* $R*$ **by** *blast*

In the case where $n = n' + 1$ for some n', we know the premisses are empty, and every premiss is derivable at a height at most n':

case (*Suc n'*)
then have (Γ $\Rightarrow*$ Δ $\oplus$ *Compound F Fs*,n'+*1*) $\in$ *derivable* $R*$ **using** a' **by** *simp*
then obtain *Ps* **where** (*Ps*, Γ $\Rightarrow*$ Δ $\oplus$ *Compound F Fs*) $\in$ $R*$ **and**
Ps $\neq$ [] **and**
$\forall$ $p \in$ *set Ps*. $\exists$ $n{\leq}n'$. (p,n) $\in$ *derivable* $R*$ **by** *auto*
then obtain r S **where** $r \in Ax \vee r \in R'$
and *extendRule* S r = (*Ps*, Γ $\Rightarrow*$ Δ $\oplus$ *Compound F Fs*) **by** *auto*
moreover
{assume $r \in Ax$ — Gives a contradiction
then have *fst* r = [] **apply** (*cases r*) **by** (*rule Ax.cases*) *auto*
moreover obtain x y **where** r = (x,y) **by** (*cases r*)

```
    then have x ≠ [] using ‹Ps ≠ []›
                and ‹extendRule S r = (Ps, Γ ⇒* Δ ⊕ Compound F Fs)› by auto
    ultimately have ∃ m≤n. (Γ + Γ' ⇒* Δ + Δ',m) ∈ derivable R*   by auto
   }
 moreover
   {assume r ∈ R'
    obtain ps c where r = (ps,c) by (cases r) auto
     have (rightPrincipal r (Compound F Fs)) ∨
           ¬(rightPrincipal r (Compound F Fs))
    by blast  — The formula is principal, or not
```

If the formula is principal, then $\Gamma' \Rightarrow \Delta'$ is amongst the premisses of r:

```
 {assume rightPrincipal r (Compound F Fs)
  then have (Γ' ⇒* Δ') ∈ set ps using b'        by auto
  then have extend S (Γ' ⇒* Δ') ∈ set Ps
      using ‹extendRule S r = (Ps,Γ ⇒* Δ ⊕ Compound F Fs)›
     by (simp)
  moreover  have S = (Γ ⇒* Δ)  by (cases S) auto
  ultimately have (Γ + Γ' ⇒* Δ + Δ') ∈ set Ps by (simp add:extend-def)
  then have ∃ m≤n'. (Γ + Γ' ⇒* Δ + Δ',m) ∈ derivable R*
     using ‹∀ p ∈ set Ps. ∃ n≤n'. (p,n) ∈ derivable R*› by auto
  then have ∃ m≤n. (Γ + Γ' ⇒* Δ + Δ',m) ∈ derivable R*  by (auto)
 }
```

If the formula is not principal, then it must appear in the premisses. The first two lines give a characterisation of the extension and conclusion, respectively. Then, we apply the induction hypothesis at the lower height of the premisses:

```
 {assume ¬ rightPrincipal r (Compound F Fs)
  obtain Φ Ψ where S = (Φ ⇒* Ψ) by (cases S) (auto)
  then obtain G H where c = (G ⇒* H) by (cases c) (auto)
  then have ⦇ Compound F Fs ⦈ ≠ H   — Proof omitted
 have Ψ + H = Δ ⊕ Compound F Fs
      using ‹S = (Φ ⇒* Ψ)› and ‹r = (ps,c)› and ‹c = (G ⇒* H)› by auto
  moreover from ‹r = (ps,c)› and ‹c = (G ⇒* H)›
  have H = ∅ ∨ (∃ A. H = ⦇A⦈) by auto
  ultimately have Compound F Fs :# Ψ   — Proof omitted
  then have ∃ Ψ1. Ψ = Ψ1 ⊕ Compound F Fs by  (auto)
 then obtain Ψ1 where S = (Φ ⇒* Ψ1 ⊕ Compound F Fs) by auto
 have ∀ p ∈ set Ps. (Compound F Fs :# succ p)  — Appears in every premiss
                    by  (auto)
  then have  ∀ p ∈ set Ps. ∃ Φ' Ψ' m. m≤n' ∧
                                (Φ' + Γ' ⇒* Ψ' + Δ',m) ∈ derivable R* ∧
                                p = (Φ' ⇒* Ψ' ⊕ Compound F Fs) using IH by (arith)
```

To this set of new premisses, we apply a new instance of r, with a different extension:

obtain Ps' **where** *eq*: $Ps' = map\ (extend\ (\Phi + \Gamma' \Rightarrow\!* \ \Psi 1 + \Delta'))\ ps$ **by** *auto*
have $(Ps',\Gamma + \Gamma' \Rightarrow\!* \ \Delta + \Delta') \in R*$ **by** *simp*
then have $\forall\ p \in set\ Ps'.\ \exists\ n{\leq}n'.\ (p,n) \in derivable\ R*$ **by** *auto*
then have $\exists\ m{\leq}n.\ (\Gamma + \Gamma' \Rightarrow\!* \ \Delta + \Delta',m) \in derivable\ R*$
using $\langle(Ps',\Gamma + \Gamma' \Rightarrow\!* \ \Delta + \Delta') \in R*\rangle$ **by** (*auto*)

All of the cases are now complete.

ultimately show $\exists\ m{\leq}n.\ (\Gamma + \Gamma' \Rightarrow\!* \ \Delta + \Delta',m) \in derivable\ R*$ **by** *blast*
qed

As an example, we show the left premiss of $R\wedge$ in **G3cp** is derivable at a height not greater than that of the conclusion. The two results used in the proof (`principal-means-premiss` and `rightInvertible`) are those we have previously shown:

lemma *conRInvert*:
assumes $(\Gamma \Rightarrow\!* \ \Delta \oplus (A \wedge\!* \ B),n) \in derivable\ (g3cp \cup Ax)*$
shows $\exists\ m{\leq}n.\ (\Gamma \Rightarrow\!* \ \Delta \oplus A,m) \in derivable\ (g3cp \cup Ax)*$
proof−
have $\forall\ r \in g3cp.\ rightPrincipal\ r\ (A \wedge\!* \ B) \longrightarrow (\emptyset \Rightarrow\!* \ \wr\ A\ \wr) \in set\ (fst\ r)$
using *principal-means-premiss* **by** *auto*
with *assms* **show** *?thesis* **using** *rightInvertible* **by** (*auto*)
qed

A rule is invertible iff every premiss is derivable at a height lower than that of the conclusion. A set of rules is invertible iff every rule is invertible. These definitions are easily formalised:

defs *invertible-def* : *invertible r R* ≡
$\forall\ n\ S.\ (r \in R \wedge (snd\ (extendRule\ S\ r),n) \in derivable\ R*) \longrightarrow$
$(\forall\ p \in set\ (fst\ (extendRule\ S\ r)).\ \exists\ m \leq n.\ (p,m) \in derivable\ R*)$

defs *invertible-set-def* : *invertible-set R* ≡ $\forall\ (ps,c) \in R.\ invertible\ (ps,c)\ R$

A set of multisuccedent uniprincipal rules is invertible if each rule has a different conclusion (lemma 20 on page 102, shown below without displaying the proof). **G3cp** has the unique conclusion property (as shown in section 7.2.2). Thus, **G3cp** is an invertible set of rules:

lemma *unique-to-invertible*:
assumes $R' \subseteq upRules \wedge R = Ax \cup R'$
and *uniqueConclusion R'*
shows *invertible-set R*

lemma *g3cp-invertible*:

shows *invertible-set* (*Ax* ∪ *g3cp*)
using *g3cp-uc* **and** *g3cp-upRules*
  and *unique-to-invertible*[**where** *R′*=*g3cp* **and** *R*=*Ax* ∪ *g3cp*]
by *auto*

7.3.1 Conclusions and Comparisons

For uniprincipal multisuccedent calculi, the theoretical results have been formalised. Moreover, the running example demonstrates that it is straightforward to implement such calculi and reason about them: proving invertibility now requires less than 25 lines of proof in most cases. As a direct comparison, the same invertibility results can be directly shown in the flexible framework of this section. To prove the same result (using the same *Isar* vernacular) requires over 300 lines of *Isabelle* code for each premiss. Thus, for two premiss rules, around 600 lines are needed (see appendix C for the full proof in the case of *R*∧).

7.4 Single Succedent Calculi

As stated in section 6.3, we must be careful when restricting sequents to single succedents. The same is true when we come to formalising section 6.3. If we have sequents as a pair of multisets, where the second is restricted to having size at most 1, then how does one extend the active part of $L\supset$ from **G3ip**? The left premiss will be $A\supset B \Rightarrow A$, and the extension will be $\Gamma \Rightarrow C$. The `extend` function must discard the C.

Rather than taking this route, we instead restrict to single formulae in the succedents of sequents. This raises its own problems, since now how does one represent the empty succedent? We introduce a dummy formula `Em`, which will stand for the empty formula:

datatype *'a form* = *At nat*
  | *Compound 'a 'a form list*
  | *ff*
  | *Em*

When we come to extend a sequent, say $\Gamma \Rightarrow C$, with another sequent, say $\Gamma' \Rightarrow C'$, we only "overwrite" the succedent if C is the empty formula:

defs *extend-def* : *extend forms seq* ≡ *if* (*succ seq* = *Em*)
  then (*antec forms* + *antec seq*) ⇒∗ (*succ forms*)
  else (*antec forms* + *antec seq* ⇒∗ *succ seq*)

Given this, it is possible to have right *Weakening* (lemma 10, p. 87), where we overwrite the empty formula if it appears as the succedent of the root of a derivation:

lemma *dpWeakR*:
assumes (Γ ⇒∗ *Em*,*n*) ∈ *derivable R*∗

and $R' \subseteq upRules$
and $R = Ax \cup R'$
shows $(\Gamma \Rightarrow\!* \ C,n) \in derivable\ R\!*$ — Proof omitted

Of course, if $C = Em$, then the above lemma is trivial. The burden is on the user not to "use" the empty formula as a normal formula. Lemma 11 (p. 87) can then be formalised:

lemma *rightInvertible*:
assumes $R' \subseteq upRules \wedge R = Ax \cup R'$
and $(\Gamma \Rightarrow\!* \ Compound\ F\ Fs,n) \in derivable\ R\!*$
and $\forall\ r' \in R.\ rightPrincipal\ r'\ (Compound\ F\ Fs) \longrightarrow (\Gamma' \Rightarrow\!* \ E) \in set\ (fst\ r')$
and $E \neq Em$
shows $\exists\ m{\leq}n.\ (\Gamma\ {+}\Gamma' \Rightarrow\!* \ E,m) \in derivable\ R\!*$

lemma *leftInvertible*:
assumes $R' \subseteq upRules \wedge R = Ax \cup R'$
and $(\Gamma \oplus Compound\ F\ Fs \Rightarrow\!* \ \delta,n) \in derivable\ R\!*$
and $\forall\ r' \in R.\ leftPrincipal\ r'\ (Compound\ F\ Fs) \longrightarrow (\Gamma' \Rightarrow\!* \ Em) \in set\ (fst\ r')$
shows $\exists\ m{\leq}n.\ (\Gamma\ {+}\Gamma' \Rightarrow\!* \ \delta,m) \in derivable\ R\!*$

G3ip can be expressed in this formalism:

inductive-set *g3ip*
where
  conL: $([\wr\ A\ \int + \wr\ B\ \int \Rightarrow\!* \ Em], \wr\ A \wedge\!* \ B\ \int \Rightarrow\!* \ Em) \in g3ip$
| *conR*: $([\emptyset \Rightarrow\!* \ A, \emptyset \Rightarrow\!* \ B], \emptyset \Rightarrow\!* \ (A \wedge\!* \ B)) \in g3ip$
| *disL*: $([\wr\ A\ \int \Rightarrow\!* \ Em, \wr\ B\ \int \Rightarrow\!* \ Em], \wr\ A \vee\!* \ B\int \Rightarrow\!* \ Em) \in g3ip$
| *disR1*: $([\emptyset \Rightarrow\!* \ A], \emptyset \Rightarrow\!* \ (A \vee\!* \ B)) \in g3ip$
| *disR2*: $([\emptyset \Rightarrow\!* \ B], \emptyset \Rightarrow\!* \ (A \vee\!* \ B)) \in g3ip$
| *impL*: $([\wr\ A \supset B\ \int \Rightarrow\!* \ A, \wr\ B\ \int \Rightarrow\!* \ Em], \wr\ (A \supset B)\ \int \Rightarrow\!* \ Em) \in g3ip$
| *impR*: $([\wr\ A\ \int \Rightarrow\!* \ B], \emptyset \Rightarrow\!* \ (A \supset B)) \in g3ip$

As expected, $R{\supset}$ can be shown invertible:

lemma *impRInvert*:
assumes $(\Gamma \Rightarrow\!* \ (A \supset B),\ n) \in derivable\ (Ax \cup g3ip)\!*$ **and** $B \neq Em$
shows $\exists\ m{\leq}n.\ (\Gamma \oplus A \Rightarrow\!* \ B,\ m) \in derivable\ (Ax \cup g3ip)\!*$
proof−
  have $\forall\ r \in (Ax \cup g3ip).\ rightPrincipal\ r\ (A \supset B) \longrightarrow$
    $(\wr A\int \Rightarrow\!* \ B) \in set\ (fst\ r)$
  proof− — Showing that $A \Rightarrow B$ is a premiss of every rule with $A{\supset}B$ principal
   {fix r
    assume $r \in (Ax \cup g3ip)$
    moreover assume $rightPrincipal\ r\ (A \supset B)$
    ultimately have $r \in g3ip$ **by** *auto* — If $A{\supset}B$ was principal, then $r \notin Ax$

```
  from ‹rightPrincipal r (A ⊃ B)› have snd r = (∅ ⇒* (A ⊃ B)) by auto
  with ‹r ∈ g3ip› and ‹rightPrincipal r (A ⊃ B)›
    have r = ([⦇A⦈ ⇒* B], ∅ ⇒* (A⊃B)) by (rule g3ip.cases) auto
  then have (⦇A⦈ ⇒* B) ∈ set (fst r) by auto
 }
 thus ?thesis by auto
 qed
with assms show ?thesis using rightInvertible by auto
qed
```

7.5 Modal Calculi

Some new techniques are needed when formalising the results of section 6.4. A set of modal operators must index formulae (and sequents and rules), there must be a method for modalising a multiset of formulae and we need to be able to handle IW rules.

The first of these is easy; instead of indexing formulae by a single type variable, we index on a pair of type variables, one which contains the propositional connectives, and one which contains the modal operators:

```
datatype ('a, 'b) form = At nat
                       | Compound 'a ('a, 'b) form list
                       | Modal 'b ('a, 'b) form list
                       | ff
```

Modalising multisets is relatively straightforward. We use the notation $! \cdot \Gamma$, where ! is a modal operator and Γ is a multiset of formulae:

```
defs modaliseMultiset-def:
 (a :: 'b) · (Γ :: ('a,'b) form multiset) ≡ {# Modal a [p]. p :# Γ #}
```

Two new rule sets are created. The first are the normal modal rules:

```
inductive-set modRules2 where
  ⟦ ps ≠ [] ; mset c = ⦇ Modal M Ms ⦈ ⟧ ⟹ (ps,c) ∈ modRules2
```

The second are the modalised context rules (definition 20, p. 89). Taking a subset of the normal modal rules, we extend using a pair of modalised multisets for context. We create a new inductive rule set called `p-e`, for "prime extend", which takes a set of modal active parts and a pair of modal operators (say ! and •), and returns the set of active parts extended with $! \cdot \Gamma \Rightarrow \bullet \cdot \Delta$:

```
inductive-set p-e :: ('a,'b) rule set ⇒ 'b ⇒ 'b ⇒ ('a,'b) rule set
 for R :: ('a,'b) rule set and M N :: 'b
```

where
$[\![r \in R \,;\, R \subseteq \mathit{modRules2} \,]\!] \Longrightarrow \mathit{extendRule}\ (M{\cdot}\Gamma \Rightarrow\!* N{\cdot}\Delta)\ r \in \mathit{p\text{-}e}\ R\ M\ N$

To encode the condition "all modalised context rules are IW rules", we need a method for extending the conclusion of a rule without extending the premisses. Again, this is simple:

defs *extendConc-def*: $\mathit{extendConc}\ S\ r \equiv (\mathit{fst}\ r,\ \mathit{extend}\ S\ (\mathit{snd}\ r))$

The extension of a rule set is now more complicated; the inductive definition has four clauses, depending on the type of rule:

inductive-set $\mathit{ext} :: ('a,'b)\ \mathit{rule\ set} \Rightarrow ('a,'b)\ \mathit{rule\ set} \Rightarrow 'b \Rightarrow 'b \Rightarrow ('a,'b)\ \mathit{rule\ set}$
for $R\ R' :: ('a,'b)\ \mathit{rule\ set}$ **and** $M\ N :: 'b$
where
ax: $[\![r \in R \,;\, r \in \mathit{Ax} \,]\!] \Longrightarrow \mathit{extendRule\ seq}\ r \in \mathit{ext}\ R\ R'\ M\ N$
| up: $[\![r \in R \,;\, r \in \mathit{upRules}]\!] \Longrightarrow \mathit{extendRule\ seq}\ r \in \mathit{ext}\ R\ R'\ M\ N$
| $\mathit{mod1}$: $[\![r \in \mathit{p\text{-}e}\ R'\ M\ N \,;\, r \in R \,]\!] \Longrightarrow \mathit{extendConc\ seq}\ r \in \mathit{ext}\ R\ R'\ M\ N$
| $\mathit{mod2}$: $[\![r \in R \,;\, r \in \mathit{modRules2} \,]\!] \Longrightarrow \mathit{extendRule\ seq}\ r \in \mathit{ext}\ R\ R'\ M\ N$

Note the new rule set carries information about which set contains the modalised context rules and which modal operators extend those prime parts.

We have two different inversion lemmata, depending on whether the rule was a modalised context rule, or some other kind of rule. We only show the former, since the latter is much the same as earlier proofs. The interesting cases are picked out:

lemma *rightInvert*:
fixes $\Gamma\ \Delta :: ('a,'b)\ \mathit{form\ multiset}$
assumes *rules*: $\mathit{R1} \subseteq \mathit{upRules} \wedge \mathit{R2} \subseteq \mathit{modRules2} \wedge \mathit{R3} \subseteq \mathit{modRules2} \wedge$
$R = \mathit{Ax} \cup \mathit{R1} \cup (\mathit{p\text{-}e}\ \mathit{R2}\ \mathit{M1}\ \mathit{M2}) \cup \mathit{R3} \wedge$
$R' = \mathit{Ax} \cup \mathit{R1} \cup \mathit{R2} \cup \mathit{R3}$
and a: $(\Gamma \Rightarrow\!* \Delta \oplus \mathit{Modal}\ M\ \mathit{Ms},n) \in \mathit{derivable}\ (\mathit{ext}\ R\ \mathit{R2}\ \mathit{M1}\ \mathit{M2})$
and b: $\forall\ r' \in R'.\ \mathit{rightPrincipal}\ r'\ (\mathit{Modal}\ M\ \mathit{Ms})\ R' \longrightarrow (\Gamma' \Rightarrow\!* \Delta') \in \mathit{set}\ (\mathit{fst}\ r')$
and *neq*: $\mathit{M2} \neq M$
shows $\exists\ m{\leq}n.\ (\Gamma + \Gamma' \Rightarrow\!* \Delta + \Delta', m) \in \mathit{derivable}\ (\mathit{ext}\ R\ \mathit{R2}\ \mathit{M1}\ \mathit{M2})$

This is the case where the last inference was a normal modal inference:

{assume $r \in \mathit{modRules2}$
obtain *ps c* **where** $r = (ps,c)$ **by** $(\mathit{cases}\ r)$ *auto*
with $\langle r \in \mathit{modRules2} \rangle$ **obtain** *T Ts* **where** $c = (\emptyset \Rightarrow\!* \wr\ \mathit{Modal}\ T\ \mathit{Ts}\ \wr) \vee$
$c = (\wr\ \mathit{Modal}\ T\ \mathit{Ts} \wr \Rightarrow\!* \emptyset)$
using *modRule2Characterise*[**where** $\mathit{Ps}{=}ps$ **and** $C{=}c$] **by** *auto*
moreover
{assume $c = (\emptyset \Rightarrow\!* \wr\ \mathit{Modal}\ T\ \mathit{Ts}\ \wr)$
then have *bb*: *rightPrincipal r* $(\mathit{Modal}\ T\ \mathit{Ts})\ R'$ **using** $\langle r = (ps,c) \rangle$ **and** $\langle r \in R \rangle$

```
    proof−
```

We need to know $r \in R$ so that we can extend the active part

```
  from ‹c = (∅ ⇒* ⦇Modal T Ts⦈)› and
       ‹r = (ps,c)› and
       ‹r ∈ R› and
       ‹r ∈ modRules2›
    have (ps,∅ ⇒* ⦇Modal T Ts⦈) ∈ R by auto
 with rules have (ps,  ∅ ⇒* ⦇Modal T Ts⦈) ∈ p-e R2 M1 M2 ∨
      (ps,  ∅ ⇒* ⦇Modal T Ts⦈) ∈ R3  by auto
 moreover
   {assume (ps, ∅ ⇒* ⦇Modal T Ts⦈) ∈ R3
    then have (ps, ∅ ⇒* ⦇Modal T Ts⦈) ∈ R' using rules by auto
   }
 moreover
   {assume (ps,∅ ⇒* ⦇Modal T Ts⦈) ∈ p-e R2 M1 M2
```

In this case, we show that Δ' and Γ' must be empty. The details are generally suppressed:

```
 then obtain Γ' Δ' r'
 where aa: (ps, ∅ ⇒* ⦇Modal T Ts⦈) = extendRule (M1·Γ' ⇒* M2·Δ') r'
         ∧ r' ∈ R2 by auto
    then have M1·Γ' = ∅  and M2·Δ' = ∅
    by (auto simp add:modaliseMultiset-def)
```

The other interesting case is where the last inference was a modalised context inference:

```
 {assume ba: r ∈ p-e R2 M1 M2 ∧
       extendConc S r = (Ps,  Γ ⇒* Δ ⊕ Modal M Ms)
 with rules obtain F Fs Γ'' Δ'' ps r' where
      ca: r = extendRule (M1·Γ'' ⇒* M2·Δ'') r' and
      cb: r' ∈ R2 and
    cc:  r' = (ps, ∅ ⇒* ⦇Modal F Fs⦈) ∨ r' = (ps,⦇Modal F Fs⦈ ⇒* ∅)
  by auto
 obtain Γ1 Δ1 where S = (Γ1 ⇒* Δ1) by (cases S) auto
 moreover
   {assume r' = (ps, ∅ ⇒* ⦇Modal F Fs⦈)
    with ba ca ‹S = (Γ1 ⇒* Δ1)› have
  eq1: (M1·Γ'' + Γ1 ⇒* M2·Δ'' + Δ1 ⊕ Modal F Fs) = (Γ ⇒* Δ ⊕ Modal M Ms)
         by (auto simp add:extendRule-def extend-def extendConc-def union-ac)
     then have Modal M Ms ∈ set-of (M2·Δ'') ∨
          Modal M Ms ∈ set-of Δ1 ∨
          Modal M Ms = Modal F Fs
     by auto
 moreover
```

```
    {assume Modal M Ms ∈ set-of (M2·Δ'') — Contradiction
    then have Modal M Ms :# M2·Δ'' by auto
    with neq
      have ∃ m≤n. (Γ + Γ' ⇒* Δ + Δ',m) ∈ derivable (ext R R2 M1 M2)
      by auto
   }
 moreover
   {assume Modal M Ms = Modal F Fs — The last inference is principal
   then have r' = (ps, ∅ ⇒* ⌊Modal M Ms⌋)
        using ‹r' = (ps,∅ ⇒* ⌊Modal F Fs⌋)› by simp
   with cb and rules have rightPrincipal r' (Modal M Ms) R'
        and r' ∈ R' by auto
   with b have (Γ' ⇒* Δ') ∈ set ps using ‹r' = (ps, ∅ ⇒* ⌊Modal M Ms⌋)›
      by (auto simp add:Ball-def)
    ultimately have ∃ m≤n. (Γ + Γ' ⇒* Δ+Δ',m) ∈ derivable (ext R R2 M1 M2)
    by auto
  }
moreover
  {assume Modal M Ms ∈ set-of Δ1 — Formula is in the implicit weakening
   then obtain Δ2 where Δ1 = Δ2 ⊕ Modal M Ms by blast
  from ba and rules
       have extendConc (Γ1 + Γ' ⇒* Δ2 + Δ') r ∈ (ext R R2 M1 M2) by auto
  moreover from ba and ca have fst (extendConc (Γ1 + Γ' ⇒* Δ2 + Δ') r) = Ps
          by (auto simp add:extendConc-def)
  ultimately have (Γ + Γ' ⇒* Δ + Δ',n'+1) ∈ derivable (ext R R2 M1 M2)
          by auto
  then have ∃ m≤n. (Γ + Γ' ⇒* Δ + Δ',m) ∈ derivable (ext R R2 M1 M2)
          using ‹n = Suc n'› by auto
 }
ultimately have ∃m≤n. ( Γ + Γ' ⇒* Δ + Δ', m) ∈ derivable (ext R R2 M1 M2)
          by blast
```

The other case, where the last inference was a left inference, is more straightforward, and so is omitted.

We guarantee no other rule has the same modal operator in the succedent of a modalised context rule using the condition $M \neq M_2$. Note this lemma only allows one kind of modalised context rule. In other words, it could not be applied to a calculus with the rules:

$$\frac{! \cdot \Gamma \Rightarrow A, \bullet \cdot \Delta}{\Gamma', ! \cdot \Gamma \Rightarrow \bullet A, \bullet \cdot \Delta, \Delta'} \; R_1 \qquad \frac{\bullet \cdot \Gamma \Rightarrow A, ! \cdot \Delta}{\Gamma', \bullet \cdot \Gamma \Rightarrow \bullet A, ! \cdot \Delta, \Delta'} \; R_2$$

since, if $([\emptyset \Rightarrow A], \emptyset \Rightarrow \bullet A) \in \mathcal{R}$, then $R_1 \in$ p-e $\mathcal{R}\ !\ \bullet$, whereas $R_2 \in$ p-e $\mathcal{R}\ \bullet\ !$. Similarly, we cannot have modalised context rules which have more than one modalised multiset in the antecedent or succedent of the active part. For instance:

$$\frac{! \cdot \Gamma_1, \bullet \cdot \Gamma_2 \Rightarrow A, ! \cdot \Delta_1, \bullet \cdot \Delta_2}{\Gamma', ! \cdot \Gamma_1, \bullet \cdot \Gamma_2 \Rightarrow \bullet A, ! \cdot \Delta_1, \bullet \cdot \Delta_2, \Delta'}$$

cannot belong to any `p-e` set. It would be a simple matter to extend the definition of `p-e` to take a *set* of modal operators, however this has not been done.

As an example, classical modal logic (the rules for which were given in section 6.4) can be formalised. The (modal) rules for this calculus are then given in two sets, the latter of which will be extended with $\Box \cdot \Gamma \Rightarrow \Diamond \cdot \Delta$:

inductive-set *g3mod2*
where
  diaR: ([$\emptyset$ $\Rightarrow*$ ⟅ *A* ⟆], $\emptyset$ $\Rightarrow*$ ⟅ $\Diamond$ *A* ⟆) $\in$ *g3mod2*
| *boxL*: ([⟅ *A* ⟆ $\Rightarrow*$ $\emptyset$], ⟅ $\Box$ *A* ⟆ $\Rightarrow*$ $\emptyset$) $\in$ *g3mod2*

inductive-set *g3mod1*
where
  boxR: ([$\emptyset$ $\Rightarrow*$ ⟅*A*⟆],$\emptyset$ $\Rightarrow*$ ⟅ $\Box$ *A* ⟆) $\in$ *g3mod1*
| *diaL*: ([⟅*A*⟆ $\Rightarrow*$ $\emptyset$],⟅ $\Diamond$ *A* ⟆ $\Rightarrow*$ $\emptyset$) $\in$ *g3mod1*

We then show the strong admissibility of the rule:

$$\frac{\Gamma \Rightarrow \Box A, \Delta}{\Gamma \Rightarrow A, \Delta}$$

lemma *invertBoxR*:
assumes $R = Ax \cup g3up \cup$ (*p-e g3mod1* $\Box$ $\Diamond$) $\cup$ *g3mod2*
and (Γ $\Rightarrow*$ Δ $\oplus$ ($\Box$ *A*),*n*) $\in$ *derivable* (*ext R g3mod1* $\Box$ $\Diamond$)
shows $\exists$ *m*$\leq$*n*. (Γ $\Rightarrow*$ Δ $\oplus$ *A*,*m*) $\in$ *derivable* (*ext R g3mod1* $\Box$ $\Diamond$)
proof −
 from *assms* **show** *?thesis*
 using *principal* **and** *rightInvert* **and** *g3* **by** *auto*
qed

where *principal* is the result which fulfils the principal formula conditions given in the inversion lemma, and *g3* is a result about rule sets.

7.6 Manipulating Rule Sets

In section 6.5 we showed that the removal of superfluous and redundant rules would not be harmful to invertibility. In section 6.6, we showed that removing combinable rules was

likewise not harmful to invertibility. Here, we formalise the results that the removal of such rules from a calculus $\mathcal{L}$ will create a new calculus $\mathcal{L}'$ which is equivalent. In other words, if a sequent is derivable in $\mathcal{L}$, then it is derivable in $\mathcal{L}'$. The results formalised in this section are for uniprincipal multisuccedent calculi.

When dealing with lists of premisses, a rule R with premisses P will be redundant given a rule R' with premisses P' if there exists some p such that $P = p\#P'$. There are other ways in which a rule could be redundant; say if $P = Q@P'$, or if $P = P'@Q$, and so on. The order of the premisses is not really important, since the formalisation operates on the finite set based upon the list. The more general "append" lemma could be proved from the lemma we give; we prove the inductive step case in the proof of such an append lemma. This is a height preserving transformation. Some of the proof is shown:

lemma *removeRedundant*:
assumes $r1 = (p\#ps,c) \wedge r1 \in upRules$
and $r2 = (ps,c) \wedge r2 \in upRules$
and $R1 \subseteq upRules \wedge R = Ax \cup R1$
and $(T,n) \in derivable\ (R \cup \{r1\} \cup \{r2\})*$
shows $\exists\ m{\leq}n.\ (T,m) \in derivable\ (R \cup \{r2\})*$
proof (*induct n rule*:*nat-less-induct*)
case *0*
have $(T,0) \in derivable\ (R \cup \{r1\} \cup \{r2\})*$ **by** *simp*
then have $([],T) \in (R \cup \{r1\} \cup \{r2\})*$ **by** (*cases*) *auto*
then obtain *S r* **where** *ext*: *extendRule S r* $= ([],T)$ **and**
$r \in (R \cup \{r1\} \cup \{r2\})$ **by** (*rule extRules.cases*) *auto*
then have $r \in R \vee r = r1 \vee r = r2$ **using** *c* **by** *auto*

It cannot be the case that $r = r_1$ or $r = r_2$, since those are uniprincipal rules, whereas anything with an empty set of premisses must be an axiom. Since $\mathcal{R}$ contains the set of axioms, so will $\mathcal{R} \cup r_2$:

then have $r \in (R \cup \{r2\})$ **using** *c* **by** *auto*
then have $(T,0) \in derivable\ (R \cup \{r2\})*$ **by** *auto*
then show $\exists\ m{\leq}n.\ (T,m) \in derivable\ (R \cup \{r2\})*$ **using** ‹*n*=*0*› **by** *auto*
next
case (*Suc n'*)
have $(T,n'+1) \in derivable\ (R \cup \{r1\} \cup \{r2\})*$ **by** *simp*
then obtain *Ps* **where** *e*: $Ps \neq []$
and *f*: $(Ps,T) \in (R \cup \{r1\} \cup \{r2\})*$
and *g*: $\forall\ P \in set\ Ps.\ \exists\ m{\leq}n'.\ (P,m) \in derivable\ (R \cup \{r1\} \cup \{r2\})*$
by *auto*
have *g'*: $\forall\ P \in set\ Ps.\ \exists\ m{\leq}n'.\ (P,m) \in derivable\ (R \cup \{r2\})*$
from *f* **obtain** *S r* **where** *ext*: *extendRule S r* $= (Ps,T)$
and $r \in (R \cup \{r1\} \cup \{r2\})$ **by** (*rule extRules.cases*) *auto*
then have $r \in (R \cup \{r2\}) \vee r = r1$ **by** *auto*

Either r is in the new rule set or r is the redundant rule. In the former case, there is nothing to do:

assume $r \in (R \cup \{r2\})$
  then have $(Ps,T) \in (R \cup \{r2\})*$ **by** *auto*
  with g' **have** $(T,n) \in$ *derivable* $(R \cup \{r2\})*$ **using** $\langle n = Suc\ n' \rangle$ **by** *auto*

In the latter case, the last inference was redundant. Therefore the premisses, which are derivable at a lower height than the conclusion, contain the premisses of r_2 (these premisses are `extend S ps`). This completes the proof:

  assume $r = r1$
  with *ext* **have** *map* (*extend S*) $(p \# ps) = Ps$ **using** a **by** (*auto*)
  then have $\forall\ P \in$ *set* (*map* (*extend S*) (*p*#*ps*)).
      $\exists\ m \leq n'.\ (P,m) \in$ *derivable* $(R \cup \{r2\})*$
    using g' **by** *simp*
  then have *h*: $\forall\ P \in$ *set* (*map* (*extend S*) *ps*).
      $\exists\ m \leq n'.\ (P,m) \in$ *derivable* $(R \cup \{r2\})*$ **by** *auto*

Recall that to remove superfluous rules, we must know that *Cut* is admissible in the original calculus (see lemma 15, p. 93). Again, we add the two distinguished premisses at the head of the premiss list; general results about permutation of lists will achieve a more general result. Even so, the following result is a special case where a single *Cut* can be used to give the empty sequent. Since one uses *Cut* in the proof, this will in general not be height-preserving:

lemma *removeSuperfluous*:
assumes $r1 = ((\emptyset \Rightarrow* \wr A \wr) \# ((\wr A \wr \Rightarrow* \emptyset) \# ps),c) \wedge r1 \in upRules$
and $R1 \subseteq upRules \wedge R = Ax \cup R1$
and $(T,n) \in$ *derivable* $(R \cup \{r1\})*$
and *CA*: $\forall\ \Gamma\ \Delta\ A.\ ((\Gamma \Rightarrow* \Delta \oplus A) \in derivable'\ R* \longrightarrow$
  $(\Gamma \oplus A \Rightarrow* \Delta) \in derivable'\ R*) \longrightarrow$
  $(\Gamma \Rightarrow* \Delta) \in derivable'\ R*$
shows $T \in derivable'\ R*$

Combinable rules can also be removed (lemma 18, p. 97). We encapsulate the combinable criterion by saying that if $(p\#P,T)$ and $(q\#P,T)$ are rules in a calculus, then we get an equivalent calculus by replacing these two rules by $((\text{extend } p\ q)\#P,T)$. Since the `extend` function is commutative, the order of p and q in the new rule is not important. This transformation is height preserving:

lemma *removeCombinable*:
assumes *a*: $r1 = (p \# ps,c) \wedge r1 \in upRules$
and *b*: $r2 = (q \# ps,c) \wedge r2 \in upRules$

and c: $r3 = (extend\ p\ q\ \#\ ps,\ c) \wedge r3 \in upRules$
and d: $R1 \subseteq upRules \wedge R = Ax \cup R1$
and $(T,n) \in derivable\ (R \cup \{r1\} \cup \{r2\})*$
shows $(T,n) \in derivable\ (R \cup \{r3\})*$

7.7 Conclusions

The results from the preceding chapters have been formalised. Since these results were new, the formalisation is new.

The formalisation is not exactly faithful. It was proved equivalent in the presence of some sane criteria, namely that one can decide which rules are normal and which are IW. For all sections except section 7.5 only normal rules were considered. The difference between metaformulae (and other metanotions) and *Isabelle* schematic variables is slight. In section 7.5, a restricted language was considered: we could not give use the full generality of section 6.4.

Only a portion of the formalisation was shown; a variety of intermediate lemmata were not made explicit. This was necessary, for the *Isabelle* theory files run to almost 8000 lines, but it is available in full at [Chapman, 2009]. However, these files do not have to be replicated for each new calculus. It takes very little effort to define a new calculus. Furthermore, proving invertibility is now a quick process; less than 25 lines of proof in most cases. As a direct comparison, the invertibility results of section 4.4 (using the rigid[3] calculus definitions) can be directly shown in the flexible framework of section 7.2. To prove the same result (using the same *Isar* vernacular) requires over 600 lines of *Isabelle* code (see appendix C for the full proof).

[3] Rigid in the sense that it cannot be reused.

Chapter 8

Conclusions

8.1 Applicability to the Field

The most obvious beneficiaries of this work are those working in an area closely related to that of this thesis. Either the theoretical results and framework could be re-used for other purposes, or the *Isabelle* files could be used for a new formalisation of some result. The best case scenario would be for someone who wanted to formalise some proof theory, but thought it was unfeasible, reading this thesis and using the techniques contained herein.

The formal proofs, with their detailing of all cases, also serve as a pedagogical aid for both students of proof theory and students of formalisation. Just as writing coherent and elegant proofs should be the aim of every mathematician, so too the goal of the formaliser should be to write coherent and elegant formal proofs. Whether that has been achieved here is for each reader to decide, but either way something can be gained from reading the proofs.

8.2 Future Directions

8.2.1 More Complex Calculi

The most obvious area of future study is tweaking, or perhaps reformulating, the criteria laid down in section 5.5 so that more exotic calculi can be analysed using the methods presented here. A different structure, such as sets or lists, could be considered for the contexts of rules. Only context-sharing rules are permitted at present. Some calculi, however, contain both context-sharing and context-splitting rules, notably the linear logic of [Girard, 1987]. Investigation of the invertibility of such rules would be a worthwhile avenue down which one could venture. The danger, of course, is that the criteria for both classifying the calculus itself, and for providing sufficient conditions for invertibility within that calculus, become too convoluted. One must avoid the situation where it is more straightforward to check

invertibility directly rather than checking a series of conditions which prove invertibility indirectly.

The first choice for extension would be first-order calculi. In order to define such calculi in a similar fashion to section 5.5, we would require substitution to be defined for metasequents and other meta notions. It is possible to avoid this definitional approach if we instead think of a rule as the infinite set of its inferences. Then, substitution needs only to be defined for the formulae and multisets. However, this would then no longer be an extension of section 5.5. Of course, one could rework those sections to follow this alternative approach. Another problem associated with first-order calculi is with formalisation. The formulae, and hence also the sequents and rules, in chapter 7 are indexed by type variables. Currently this is not possible in *Nominal Isabelle* (see appendix B).

Some calculi, such as **hGKi**, the Gentzen-Kleene calculus for intuitionistic logic with head formulae, have additional structure on the left of the sequent arrow in the form of a *stoup* or head. A reserved place, which can be filled by a formula, highlights the principal formula of a left inference. This extra structure simplifies analysis: if we come across a derivation which ends with:

$$A_0 \supset A_1; \Gamma \Rightarrow B$$

then this can only have had $L\supset$ or a right rule as the last inference used. There is the additional structural rule called *Dereliction*, which moves a formula from the context to the stoup position. To prove invertibility, one may have to use instances of *Dereliction*, which will make strong invertibility unlikely.

Interestingly, the left premiss of the rule $L\supset$ is not invertible for **hGKi**, but *for a different reason* to why it is not invertible in **G3i**. In **G3i**, the base case of the proof by induction failed: there is no reason $\Gamma \oplus A_0 \supset A_1 \Rightarrow B$ should be derivable for general Γ and B. However, with stoups the base case is treated by *ex falso*: if the derivation ends with $A_0 \supset A_1; \Gamma \Rightarrow B$ then it *cannot* also end with $P; \Gamma \Rightarrow P$, where P is some propositional variable. The proof now fails because of the interplay between $R\supset$ and $L\supset$. When the last inference used is $R\supset$, the derivation ends with:

$$\frac{A_0 \supset A_1; \Gamma, B \Rightarrow C}{A_0 \supset A_1; \Gamma \Rightarrow B \supset C}$$

We can apply the induction hypothesis to the premiss, but this gives a copy of B in the context which is unwanted and impossible to remove:

$$\frac{A_0 \supset A_1; \Gamma, B \Rightarrow C}{-; \Gamma, B, A_0 \supset A_1 \Rightarrow A_0}\ ih$$

More study is needed to identify which combinations of rules, or rather which properties of rules, will create the above situation so that it can be restricted.

8.2.2 Other Formalisms

The next area of interest is that of other formalisms. Sequent calculus has its roots with Gentzen in the 1930s. Since then, many formalisms have sprung up to answer questions which the sequent calculus was unable to answer. Such things include proof nets [Girard, 1987] and deep inference [Brünnler, 2006]. Both were developed to remove the redundancy present in sequent calculi, however invertibility-type questions can still be asked about such systems.

8.2.3 Related Problems

Another option one could investigate is to widen the scope from invertibility to permutability. Permutability is a more general problem, and is concerned with when two inferences can be switched without altering the provability of a sequent. For example, in **G3cp**, it is possible to switch the order of the $L\wedge$ and $R\vee$ inferences in the following derivations:

$$\dfrac{\dfrac{\Gamma, A, B \Rightarrow A, B, \Delta}{\Gamma, A \wedge B \Rightarrow A, B, \Delta}\, L\wedge}{\Gamma, A \wedge B \Rightarrow A \vee B, \Delta}\, R\vee \qquad \dfrac{\dfrac{\Gamma, A, B \Rightarrow A, B, \Delta}{\Gamma, A, B \Rightarrow A \vee B, \Delta}\, R\vee}{\Gamma, A \wedge B \Rightarrow A \vee B, \Delta}\, L\wedge$$

In this case, we say $L\wedge$ *permutes over* $R\vee$. This relates to invertibility in the following way: if a rule can be permuted over any other rule, then the initial rule is invertible. Thus, invertibility is a special kind of permutability. Giving conditions for when two rules can be permuted over one another will then provide lemma 8 as a corollary, for instance.

Permutability has been shown for specific logics, but not in a general setting. For instance, [Kleene, 1952] shows which inferences in **LK** and **LJ** can be permuted, which is extended in [Dyckhoff and Pinto, 1999] and [Shankar, 1992]. In [Troelstra and Schwichtenberg, 2000], the one-sided Gentzen-Schütte system **GS1** is analysed for permutability.

In a similar vein to permutability, albeit one further removed from the original area of study, one may investigate other meta-theoretical properties of sequent calculi. The prime candidate is that of *Cut* admissibility, which has been studied in some detail in [Ciabattoni and Terui, 2006a], [Curry, 1963], [Restall, 1999] etc. Indeed, looking at the conditions of [Restall, 1999] when compared with the defnition of a decomposable uniprincipal calculus (section 5.5) one sees all that is missing for such a calculus to admit *Cut* is the matching of principal constituents. In other words, should one provide conditions for when a cut, whose formula is principal in *both* premisses, can be eliminated in such a calculus (subject to use of the induction hypothesis), then one would have proved *Cut* admissibility in general.

Also of interest is *Contraction*. This is perhaps more accessible than *Cut* admissibility, given that only one rule at a time needs to be considered, rather than pairs of rules. However, as we saw in section 4.5, based on [Dyckhoff and Negri, 2000], a lot of non-standard lemmata are needed in some instances to prove *Contraction* admissibility. *Contraction* is also, whilst interesting, not as important as *Cut* admissibility, nor indeed is it as important as proving

an interpolation result. As noted in section 4.3, interpolation results are useful in various branches of computer science. Given a logic, would it be possible to give conditions for when an interpolation theorem is provable?

8.2.4 Formalisation Problems

From the theorem prover side, there is one avenue which would be very interesting to explore. The informal proofs given within this thesis often contain sequent calculus derivations, which the formal proof has to recreate line by line in a linear fashion. Ideally one would like to give a sequent calculus derivation in *Isabelle* using the same mechanism as one would in LaTeX, with some notes to tell the system which rule to apply etc. Then, instead of getting formally checked proofs, one would get formally checked proofs which contain formally checked sequent calculus derivations. This would, again, decrease the mismatch between the formal and informal approaches. Incidentally, this was one of the original ideas for the thesis which was never followed through; it is more a LaTeX and *Isabelle* interface problem than a proof theory problem.

8.3 Final Comments

As stated in section 1.2, the objective of the thesis was to show how human readable proofs, or the least human legible proofs, of results from structural proof theory could be formalised. The examples of chapter 4 show that, if this was not fully realised, then a large step was taken down this path. The three formalisations given in chapter 4 were novel: they showed *Isar* used for non-trivial formalisation of proof theory. The ideal would be for the *Isar* language, or a similar style vernacular in a different theorem prover, to progress to the stage where a formalised proof is indistinguishable from an informal proof.

In chapters 5-7, we gave a series of (formalised) results about invertibility which reduce the burden on the user. Of course, one is restricted by the choices made in the formalisation: multisets and context sharing rules are all that is available. These chapters contained new work: the theoretical work was novel and, when formalised, created a new framework for proof theory in *Isabelle*. Not all obstacles have been overcome, however some of the barriers to entry for formalising proof theory have been removed by this work.

It is hoped that the work contained in this thesis can be used by others. The aim of formalising as much of mathematics as possible is indeed lofty, and is more than one thesis can ever achieve. However, the growing body of formalised proofs, of which this work forms a small, imperfect part, can guide the way for others by showing that formalisation is not something which should be feared; it is not something which stands in opposition to informal mathematics, but rather can go hand in hand with it.

Bibliography

A. Adams. *Tools and Techniques for Machine-Assisted Meta-Theory*. PhD thesis, University of St. Andrews, 1997.

M. Aigner and G. M . Ziegler. *Proofs from the Book*. Springer-Verlag, 2003.

A. Avron. Simple consequence relations. *Inf. Comput.*, 92(1):105–140, 1991.

A. Avron and I. Lev. Canonical propositional gentzen-type systems. In *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, volume 2083 of *Lecture Notes in Computer Science*. Springer, 2001.

B. E. Aydemir, A. Bohannon, and S. Weirich. Nominal reasoning techniques in coq: (extended abstract). *Electr. Notes Theor. Comput. Sci.*, 174(5):69–77, 2007.

F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. Elsevier, 1981.

N. Belnap Jr. Display Logic. *Journal of Philosophical Logic*, 11:375–417, 1982.

Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004. URL http://www.labri.fr/publications/l3a/2004/BC04.

S. Boulmé. A Proof of Craig's Interpolation Theorem in Coq, 1996. URL citeseer.ist.psu.edu/480840.html.

A. Bove, P. Dybjer, and U. Norell. A brief overview of agda - functional programming with dependent types. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009, Proceedings*, volume ? of *Lecture Notes in Computer Science*, page ? Springer, 2009.

K. Brünnler. Deep inference and its normal form of derivations. In *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006,*

Swansea, UK, June 30-July 5, 2006, Proceedings, volume 3988 of *Lecture Notes in Computer Science*, pages 65–74. Springer, 2006.

P. Chapman. Invertibility in Sequent Calculi. In G. Klein and T. Nipkow and L. Paulson, editor, *The Archive of Formal Proofs*. http://afp.sourceforge.net/entries/SequentInvertibility.shtml, August 2009. Formal proof development.

P. Chapman, J. McKinna, and C. Urban. Mechanising a Proof of Craig's Interpolation Theorem for Intuitionistic Logic in Nominal Isabelle. In *AISC/MKM/Calculemus*, volume 5144 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 2008.

A. Ciabattoni and K. Terui. Towards a Semantic Characterization of Cut-Elimination. *Studia Logica*, 82(1):95–119, 2006a.

A. Ciabattoni and K. Terui. Modular Cut-Elimination: Finding Proofs or Counterexamples. In *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings*, pages 135–149, 2006b.

Coq Development Team. *The Coq Proof Assistant Reference Manual Version 8.1*, 2006. Available at http://coq.inria.fr/V8.1/refman/index.html.

T. Coquand and G. P. Huet. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120, 1988.

P. Corbineau. A declarative language for the coq proof assistant. In *Types for Proofs and Programs, International Conference, TYPES 2007, Cividale des Friuli, Italy, May 2-5, 2007, Revised Selected Papers*, volume 4941 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2008.

H. B. Curry. *Foundations of Mathematical Logic*. McGraw-Hill Series in Higher Mathematics. McGraw-Hill Book Company, 1963.

J. E. Dawson and R. Goré. Embedding display calculi into logical frameworks: Comparing twelf and isabelle. *Electr. Notes Theor. Comput. Sci.*, 42, 2001.

J. E. Dawson and R. Goré. Formalised cut admissibility for display logic. In *Theorem Proving in Higher Order Logics, 15th International Conference, TPHOLs 2002, Hampton, VA, USA, August 20-23, 2002, Proceedings*, volume 2410 of *Lecture Notes in Computer Science*, pages 131–147. Springer, 2002.

J.E. Dawson. Isabelle files on invertibility. Available at http://users.rsise.anu.edu.au/∼jeremy/isabelle/, 2008.

N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation with application to the church-rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

A. G. Dragalin. *Mathematical Intuitionism.* Number 67 in Translations of Mathematical Monographs. American Mathematical Society, 1988.

R. Dyckhoff and S. Lengrand. Ljq: A strongly focused calculus for intuitionistic logic. In *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings*, volume 3988 of *Lecture Notes in Computer Science*, pages 173–185. Springer, 2006.

R. Dyckhoff and S. Negri. Admissibility of Structural Rules for Contraction-Free Systems of Intuitionistic Logic. *J. Symb. Log.*, 65(4):1499–1518, 2000.

R. Dyckhoff and L. Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theor. Comput. Sci.*, 212(1-2):141–155, 1999.

J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax Involving Binders. In *14th Annual Symposium on Logic in Computer Science*, pages 214–224, Washington, DC, USA, 1999. IEEE Computer Society Press. ISBN 0-7695-0158-3.

G. Gentzen. *The Collected Papers of Gerhard Gentzen.* North-Holland Publishing Company, 1969.

J-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

J-Y. Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

H. Herbelin and G. Lee. Forcing-based cut-elimination for gentzen-style intuitionistic sequent calculus. In *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 209–217. Springer, 2009.

R. Hori, H. Ono, and H. Schellinx. Extending intutionistic linear logic with knotted structural rules. *Notre Dame Journal of Formal Logic*, 35(2):219–242, 1994.

R. Jhala, R. Majumdar, and R-G. Xu. State of the Union: Type Inference Via Craig Interpolation. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 553–567. Springer, 2007.

M. Kanazawa. Computing interpolants in implicational logics. *Ann. Pure Appl. Logic*, 142 (1-3):125–201, 2006.

S. C. Kleene. Permutability of inferences in Gentzen's calculi LK and LJ. *Memoirs of the American Mathematical Society*, 10:1–26, 1952.

P. Martin-Löf. *Intuitionistic Type Theory*. Number 1 in Studies in Proof Theory. Bibliopolis, 1984.

C. McBride. *Epigram: Practical Programming with dependent types*, 2005. Available at http://strictlypositive.org/publications/epigram-notes.ps.gz.

C. McBride. Epilogue for epigram. Available at http://www.e-pig.org/epilogue/, 2008.

C. McBride and J. McKinna. The View from the Left. *J. Funct. Program.*, 14(1):69–111, 2004.

J. McKinna and R. Pollack. Pure Type Systems Formalized. In M. Bezem and J. F. Groote, editors, *Proceedings 1st Int. Conf. on Typed Lambda Calculi and Applications, TLCA'93, Utrecht*, volume 664 of *LNCS*, pages 289–305. Springer-Verlag, 1993. URL citeseer.ist.psu.edu/mckinna93pure.html.

K. L. McMillan. Applications of Craig Interpolants in Model Checking. In *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3440 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

S. Negri. Proof Analysis in Modal Logic. *Journal of Philosophical Logic*, 34:507–544, 2005.

S. Negri and J. von Plato. *Structural Proof Theory*. Cambridge University Press, Cambridge, 2001.

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.

U. Norell. Dependently typed programming in agda. In *Proceedings of TLDI'08: 2008 ACM SIGPLAN International Workshop on Types in Languages Design and Implementation, Savannah, GA, USA, January 24, 2009*, pages 1–2. ACM, 2009.

F. Pfenning. Structural Cut Elimination: I. Intuitionistic and Classical Logic. *Inf. Comput.*, 157(1-2):84–141, 2000.

F. Pfenning and C. Paulin-Mohring. Inductively defined types in the calculus of constructions. In *Mathematical Foundations of Programming Semantics*, volume 442 of *Lecture Notes in Computer Science*, pages 209–228. Springer, 1989.

F. Pfenning and C. Schuermann. *Twelf User's Guide*, 2005.

A. M. Pitts. Nominal Logic, a First-Order Theory of Names and Binding. *Inf. Comput.*, 186(2):165–193, 2003. ISSN 0890-5401. doi: http://dx.doi.org/10.1016/S0890-5401(03)00138-X.

J. Rasga. Sufficient conditions for cut elimination with complexity analysis. *Ann. Pure Appl. Logic*, 149(1-3):81–99, 2007.

T. M. Rasmussen. *Interval logic. Proof theory and theorem proving.* PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2002.

G. Restall. *An Introduction to Substructural Logics.* Routledge, London, 1999.

T. Ridge. Craig's Interpolation Theorem formalised and mechanised in Isabelle/HOL. Arxiv preprint cs.LO/0607058, 2006 - arxiv.org, 2006.

C. Schürmann. *Automating the Meta Theory of Deductive Systems.* PhD thesis, Carnegie Mellon University, 2000.

K. Schütte. *Proof Theory.* Springer-Verlag, 1977.

N. Shankar. Proof search in the intuitionistic sequent calculus. In *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*, volume 607 of *Lecture Notes in Computer Science*, pages 522–536. Springer, 1992.

D.J. Shoesmith and T. J. Smiley. *Multiple-Conclusion Logic.* Cambridge University Press, 2008.

O. Sonobe. A Gentzen-type formulation of some intermediate propositional logics. *Journal of Tsuda College*, 7, 1975.

G. Takeuti. *Proof Theory.* Number 81 in Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Company, 1975.

C. Tasson and C. Urban. Nominal techniques in Isabelle/HOL. In *Proceedings of the 20th International Conference on Automated Deduction (CADE 2005)*, volume 3632 of *LNCS*, pages 38–53. Springer-Verlag, 2005.

Mizar team. Mizar website. Available at http://www.mizar.org/, 2008.

A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Number 43 in Cambridge Tracts in Computer Science. Cambridge University Press, second edition, 2000.

C. Urban. *Classical Logic and Computation.* PhD thesis, Cambridge University, 2000.

C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundam. Inform.*, 45(1-2):123–155, 2001.

C. Urban and B. Zhu. Revisiting cut-elimination: One difficult proof is really a proof. In *Rewriting Techniques and Applications, 19th International Conference, RTA 2008, Hagenberg, Austria, July 15-17, 2008, Proceedings*, volume 5117 of *Lecture Notes in Computer Science*, pages 409–424. Springer, 2008.

C. Urban, S. Berghofer, and M. Norrish. Barendregt's variable convention in rule inductions. In *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2007.

M. Wenzel. *Isabelle/Isar Reference Manual*, 2002.

M. Wenzel. Structured induction proofs in isabelle/isar. In *Mathematical Knowledge Management, 5th International Conference, MKM 2006, Wokingham, UK, August 11-12, 2006, Proceedings*, volume 4108 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2006.

A. Zamansky and A. Avron. Canonical Gentzen-Type Calculi with (n, k)-ary Quantifiers. In *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, pages 251–265, 2006.

Appendix A

Major Proof Systems

In this appendix, we will give a background and description of the major proof systems used in this thesis.

A.1 G3cp

Gentzen initially created two sequent calculi: one for classical logic (**LK**) and one for intuitionistic logic (**LJ** or **LI**). The calculus **G3cp** is a descendent of **LK**: it is a calculus for propositional classical logic, which uses multisets on both the left and right of the sequent arrow. Although one can interdefine the connectives in classical logic, in this thesis, and indeed in most presentations of **G3cp**, we take all the connectives as primitive. We also have $\bot$ as a distinguished propositional atom. The rules are given in context-sharing form. The rules are as follows:

$$\frac{}{\Gamma, P \Rightarrow \Delta, P}\ Ax \qquad \frac{}{\Gamma, \bot \Rightarrow \Delta}\ L\bot$$

$$\frac{\Gamma \Rightarrow \Delta, \phi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \phi \wedge \psi}\ R\wedge \qquad \frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}\ L\wedge$$

$$\frac{\Gamma \Rightarrow \Delta, \phi, \psi}{\Gamma \Rightarrow \Delta, \phi \vee \psi}\ R\vee \qquad \frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}\ L\vee$$

$$\frac{\Gamma, \phi \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \phi \supset \psi}\ R\supset \qquad \frac{\Gamma \Rightarrow \Delta, \phi \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \supset \psi \Rightarrow \Delta}\ L\supset$$

A.2 G3ip

G3ip is a calculus for intuitionistic logic, based upon Gentzen's original **LJ**. It is similar to **G3cp**, but there is a restriction that the succedent must contain at most one formula.

This means any rule with two distinguished formulae on the right for **G3cp** must be altered. This is why there are two rules for disjunction on the right. In order to deal with the non-invertibility of $L\supset$, we must copy the principal formula of a rule into the premisses for that rule. Unfortunately, this creates the problem of looping during proof search: we can apply this rule over and over again. In intuitionistic logic, connectives cannot be defined in terms of the other connectives. Thus we have to give rules for all connectives, rather than doing so for notational convenience. Again, we use multisets for contexts:

$$\frac{}{\Gamma, P \Rightarrow P}\ Ax \qquad \frac{}{\Gamma, \bot \Rightarrow \delta}\ L\bot$$

$$\frac{\Gamma \Rightarrow \phi \quad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \wedge \psi}\ R\wedge \qquad \frac{\Gamma, \phi, \psi \Rightarrow \delta}{\Gamma, \phi \wedge \psi \Rightarrow \delta}\ L\wedge$$

$$\frac{\Gamma, \phi \Rightarrow \psi}{\Gamma \Rightarrow \phi \supset \psi}\ R\supset \qquad \frac{\Gamma, \phi \supset \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \delta}{\Gamma, \phi \supset \psi \Rightarrow \delta}\ L\supset$$

$$\frac{\Gamma \Rightarrow \phi}{\Gamma \Rightarrow \phi \vee \psi}\ R\vee_1 \qquad \frac{\Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \vee \psi}\ R\vee_2$$

$$\frac{\Gamma, \phi \Rightarrow \delta \quad \Gamma, \psi \Rightarrow \delta}{\Gamma, \phi \vee \psi \Rightarrow \delta}\ L\vee$$

A.3 G4ip

When performing proof search using **G3ip** one encounters a problem. Because the principal formula of $L\supset$ is copied into the left premiss, we can apply that rule over and over again. Thus, it is possible for proof search to loop. If we wish to automate proof search, we must then implement a loop-checker, which is inefficient. The calculus **G4ip** was developed to remove the need for a loop-checker: instead of one rule for implication on the left, we analyse the structure of the antecedent of the implication and apply different rules accordingly. The rules for the other connectives and the rule $R\supset$ are the same as those for **G3ip**, whereas the four rules for implication on the left are:

$$\frac{\Gamma, P, \psi \Rightarrow \delta}{\Gamma, P, P \supset \psi \Rightarrow \delta}\ L0\supset \qquad \frac{\Gamma, \alpha \supset (\beta \supset \psi) \Rightarrow \delta}{\Gamma, (\alpha \wedge \beta) \supset \psi \Rightarrow \delta}\ L\wedge\supset$$

$$\frac{\Gamma, \alpha \supset \psi, \beta \supset \psi \Rightarrow \delta}{\Gamma, (\alpha \vee \beta) \supset \psi \Rightarrow \delta}\ L\vee\supset \qquad \frac{\Gamma, \alpha, \beta \supset \psi \Rightarrow \beta \quad \Gamma, \psi \Rightarrow \delta}{\Gamma, (\alpha \supset \beta) \supset \psi \Rightarrow \delta}\ L\supset\supset$$

A.4 G3i

It is possible to add rules for quantifiers, so that we have a calculus for first-order intuitionistic logic. The propositional rules are the same as those for **G3ip**, and we add the four

rules:

$$\frac{\Gamma \Rightarrow [t/x]A}{\Gamma \Rightarrow \exists x.A}\ R\exists \qquad \frac{\Gamma, \exists x.A, [y/x]A \Rightarrow \delta}{\Gamma, \exists x.A \Rightarrow \delta}\ L\exists$$

$$\frac{\Gamma \Rightarrow [y/x]A}{\Gamma \Rightarrow \forall x.A}\ R\forall \qquad \frac{\Gamma, \forall x.A, [t/x]A \Rightarrow \delta}{\Gamma, \forall x.A \Rightarrow \delta}\ L\forall$$

where, in the rules $L\exists$ and $R\forall$, the variable y must not appear free in the conclusion of the rule.

A.5 G3s

We can add rules for the $S4$ modal operators $\Box$ and $\Diamond$ to **G3cp**. Again, the propositional rules are the same as for **G3cp**, and the four modal rules are:

$$\frac{\Box\Gamma \Rightarrow \phi, \Diamond\Delta}{\Gamma', \Box\Gamma \Rightarrow \Box\phi, \Diamond\Delta, \Delta'}\ R\Box \qquad \frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma, \Box\phi \Rightarrow \Delta}\ L\Box$$

$$\frac{\Gamma \Rightarrow \Delta, \phi}{\Gamma \Rightarrow \Delta, \Diamond\phi}\ R\Diamond \qquad \frac{\Box\Gamma, \phi \Rightarrow \Diamond\Delta}{\Gamma', \Box\Gamma, \Diamond\phi \Rightarrow \Diamond\Delta, \Delta'}\ L\Diamond$$

Appendix B

Personal Communication

Christian Urban is the creator and main developer of *Nominal Isabelle*. This is his response to the question of whether type variables could be used in *Nominal Isabelle*:

> In theory there is no problem with using type-variables in nominal datatypes, but they make things unwieldy. Remember in Nominal everything depends on proper definitions of permutation operations. This property is loaded onto the type-system. That means if *Isabelle* looks at a type it knows immediately whether this property holds for elements of this type.
>
> Now type variables can stand for arbitrary types. In this context, this is not wanted. What is wanted is a restriction that says the type variable stands only for types which have a proper definition of permutations. This restrictions can be achieved with annotations of the form:
>
> 'a::being-a-permutation-type
>
> This on its own does not make things too complicated, but the problems start when things scale. The reason is that unfortunately, definition of permutations is not the only property that is needed. Often one needs to know that things have finite support. This results in annotations like:
>
> 'a::being-a-permutation-type, is-finitely-supported
>
> From a glitch in the implementation of *Isabelle*, this needs to be repeated for every atom type. So if you have two atom types, you already have four constraints.
>
> The showstopper however is that one needs to know that every atomtype is independent from each other. This results in another two constraints, but grows quadratically with the number of atom types.
>
> So in the end it is just a practical problem, and I have no real idea how I can solve it given the current infrastructure.

Appendix C

A Direct Comparison

C.1 Introduction

Whilst the work of the preceding chapters is interesting in its own right, it carries more weight if we can show it reduces the burden on the user when formalising structural proof theory. This chapter contains a direct comparison; using the robust framework from chapter 7 the invertibility of $R\wedge$ is proved directly.

C.2 Comparing Invertibility Proofs

In §7.3, the invertibility of the $R\wedge$ in **G3cp** was shown (with respect to the left premiss). Recall from lemma 20 that one could invoke a different argument to show invertibility of $R\wedge$; **G3cp** has the unique conclusion property (definition 23)). Using this, the derivability at a lower height of both premisses can be shown in one lemma:

```
lemma invertConR:
assumes (Γ ⇒* Δ ⊕ (A∧*B),n) ∈ derivable (Ax ∪ g3cp)*
shows ∃ m≤n. (Γ ⇒* Δ ⊕ A,m) ∈ derivable (Ax ∪ g3cp)*
  and ∃ m≤n. (Γ ⇒* Δ ⊕ B,m) ∈ derivable (Ax ∪ g3cp)*
proof−
have extendRule (Γ ⇒* Δ) ([{#} ⇒* {#A#},{#} ⇒* {#B#}],{#} ⇒* {#A∧*B#})
                                  ∈ (Ax ∪ g3cp)* by auto
moreover have
   extendRule (Γ ⇒* Δ) ([{#} ⇒* {#A#},{#} ⇒* {#B#}],{#} ⇒* {#A∧*B#})
          =   ([Γ ⇒* Δ ⊕ A, Γ ⇒* Δ ⊕ B],Γ ⇒* Δ ⊕ (A∧*B))    by (auto)
ultimately
   have ([ Γ ⇒* Δ ⊕ A,  Γ ⇒* Δ ⊕ B],  Γ ⇒* Δ ⊕ (A∧*B)) ∈ (Ax ∪ g3cp)*
        by simp
with assms show ∃ m≤n. ( Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
```

```
15              and ∃ m≤n. ( Γ ⇒* Δ ⊕ B, m) ∈ derivable (Ax ∪ g3cp)*
16      using invertibleRule[where R'=g3cp] by (auto simp add:g3cp-uc g3cp-upRules)
17  qed
```

Even with the additional proofs of `g3cp-uc` and `g3cp-upRules`, the total amount written is under 30 lines.

The direct proof, by comparison, is long and bloated. Firstly, it does not seem possible to prove both statements within one proof. In other words, one needs separate left and right premiss proofs. Each of these consists proofs is by induction on the height of the derivation of the conclusion:

```
lemma invertConR2L:
assumes (Γ ⇒* Δ ⊕ (A ∧* B), n) ∈ derivable (Ax ∪ g3cp)*
shows ∃ m≤n. (Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
using assms
proof (induct n arbitrary: Γ Δ rule: nat-less-induct)
```

There is a trivial base case, which has two separate subcases (the proofs are suppressed:

```
case 0
with der have (Γ ⇒* Δ ⊕ (A ∧* B), 0) ∈ derivable (Ax ∪ g3cp)* by simp
then have ([], Γ ⇒* Δ ⊕ (A ∧* B)) ∈ (Ax ∪ g3cp)* by (rule derivable.cases) auto
then obtain S r where ext: extendRule S r = ([],Γ ⇒* Δ ⊕ (A ∧* B))
   and       r ∈ (Ax ∪ g3cp) by (rule extRules.cases) auto
 ultimately have r ∈ Ax by auto
then obtain i where r = ([], {#ff#} ⇒* {#}) ∨ r = ([],{#At i#} ⇒* {#At i#})
    apply (cases r) by (rule Ax.cases) auto
  ultimately have (Γ ⇒* Δ ⊕ A, 0) ∈ derivable (Ax ∪ g3cp)* by blast
then show ∃ m≤n. (Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
    using ‹n=0› by blast
next
```

When the height is a positive integer, $A \wedge B$ was either principal in the last instance, or it was not. There are the same number of non-principal cases as rules, and this is where the bloated nature of the proof arises:

```
case (Suc n')
 with der have (Γ ⇒* Δ ⊕ (A ∧* B), n'+1) ∈ derivable (Ax ∪ g3cp)* by simp
 then obtain Ps S r where nonempty: Ps ≠ []
                      and       ext': (Ps, Γ ⇒* Δ ⊕ (A ∧* B)) ∈ (Ax ∪ g3cp)*
                      and     premss: ∀ p ∈ set Ps. ∃ m≤n'. (p,m) ∈ derivable (Ax ∪ g3cp)*
     using characteriseLast by auto
 from ext' obtain S r where ext: extendRule S r = (Ps, Γ ⇒* Δ ⊕ (A ∧* B))
             and       r ∈ (Ax ∪ g3cp) by (rule extRules.cases) auto
  ultimately have r ∈ g3cp by auto
moreover obtain ps c where r = (ps,c) by (cases r) auto
```

```
 ultimately have (ps,c) ∈ g3cp by simp
then have ∃ m≤n'+1. (Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
  proof (cases) — Case analysis on the last rule used
  case (conR D E)
  have D ∧* E = A∧*B ∨ D ∧* E ≠ A ∧* B by blast
  moreover
   {assume D ∧* E = A ∧* B — The one principal case
    with ext and ‹c = ({#} ⇒* {#D∧*E#})› and ‹r = (ps,c)›
      have S = (Γ ⇒* Δ) by (cases S) auto
  with ‹ps = [{#} ⇒* {#D#}, {#} ⇒* {#E#}]› and ext
      and ‹D∧*E = A∧*B› and ‹r = (ps,c)›
      have Ps = [Γ ⇒* Δ ⊕ A, Γ ⇒* Δ ⊕ B] by (auto)
  with premss have ∃ m≤n'. (Γ ⇒* Δ ⊕ A,m) ∈ derivable (Ax ∪ g3cp)*
      by (auto)
  then have ∃ m≤n'+1. (Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
      by (rule-tac x=m in exI) auto
  }
 moreover
  {assume D ∧* E ≠ A ∧* B — One of many non-principal cases
         }
 next
    case (impR D E) — Each non-principal case is around 35 lines long
     next
     case (disR D E)
      next
     case (impL D E)
    next
    case (disL D E)
   next
    case (conL D E)
   qed
  then show ∃ m≤n. (Γ ⇒* Δ ⊕ A, m) ∈ derivable (Ax ∪ g3cp)*
      using ‹n = Suc n'› by auto
qed
```

The whole proof is around 300 lines long (depending on spacing etc.). Even if both premisses could be handled with one lemma of this length, we still have to write roughly ten times more using the direct proof. The efficiency for such proofs is not as relevant; checking 30 lines and checking 300 lines happens quickly. In fact, given that the earlier invertibility results need to be loaded beforehand, the direct proof takes less time to be checked. However, it takes much longer to write.

As is obvious, with more rules, the disparity becomes even larger between the direct and indirect proof lengths. For every rule added, one gets an extra non-principal case in the

proof. Whilst it is only a linear increase, using the indirect method there is no increase in proof length at all.

Appendix D

Rigid Formalisations

This chapter contains two of the formalisations of chapter 4, *Cut* admissibility for **G3ip** and *Contraction* admissibility for **G4ip**. The formalisation of of section 4.3 is included in the *Nominal Isabelle* distribution. The formalisations of chapter 7 are not contained here: they are available online [Chapman, 2009].

D.1 Cut Admissibility for G3ip

This file uses Multiset.thy, which is included in the *Isabelle* distribution.

datatype *form* = *Atom nat*
| *Imp form form* (-⊃- [*100*,*100*]*110*)
| *Conj form form* (- ∧∗ - [*100*,*100*]*110*)
| *Disj form form* (- ∨∗ - [*100*,*100*]*110*)
| *ff*

abbreviation
multiset-plus (**infixl** ⊕ *80*) **where**
(Γ :: *form multiset*) ⊕ (*A* :: *form*) ≡ Γ + {#*A*#}
abbreviation
multiset-minus (**infixl** ⊖ *80*) **where**
(Γ :: *form multiset*) ⊖ (*A* :: *form*) ≡ Γ − {#*A*#}

inductive
provable-dp :: *form multiset* ⇒ *form* ⇒ *nat* ⇒ *bool* (- ⇒ - ↓ - [*60*,*60*,*60*] *60*)
where
Ax[*intro*]: ⟦(*Atom i*):# Γ⟧ ⟹ Γ ⇒ *Atom i* ↓ *0*
| *LBot*[*intro*]: ⟦ *ff* :# Γ⟧ ⟹ Γ ⇒ *C* ↓ *0*
| *ConjR*[*intro*]: ⟦Γ ⇒ *A* ↓ *n* ; Γ ⇒ *B* ↓ *m*⟧ ⟹ Γ ⇒ *A* ∧∗ *B* ↓ *n*+*m*+*1*
| *ConjL*[*intro*]: ⟦Γ ⊕ *A* ⊕ *B* ⇒ *C* ↓ *n*⟧ ⟹ Γ ⊕ *A*∧∗*B* ⇒ *C* ↓ *n*+*1*

| *DisjR1*[*intro*]: $[\![\Gamma \Rightarrow A \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \vee * B \downarrow n{+}1$
| *DisjR2*[*intro*]: $[\![\Gamma \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \vee * B \downarrow n{+}1$
| *DisjL*[*intro*]: $[\![\Gamma \oplus A \Rightarrow C \downarrow n \ ; \ \Gamma \oplus B \Rightarrow C \downarrow m]\!] \Longrightarrow \Gamma \oplus A \vee * B \Rightarrow C \downarrow n{+}m{+}1$
| *ImpR*[*intro*]: $[\![\ \Gamma \oplus A \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A \supset B \downarrow n{+}1$
| *ImpL*[*intro*]: $[\![\ \Gamma \oplus A \supset B \Rightarrow A \downarrow n \ ; \ \Gamma \oplus B \ \Rightarrow C \downarrow m]\!] \Longrightarrow \Gamma \oplus A \supset B \Rightarrow C \downarrow n{+}m{+}1$

consts *length* :: *form* ⇒ *nat*
primrec
length (*Atom i*) = *0*
length (*A* ⊃ *B*) = (*if* (*length A* ≤ *length B*) *then* (*length B* + *1*) *else* (*length A* + *1*))
length (*A* ∧∗ *B*) = (*if* (*length A* ≤ *length B*) *then* (*length B* + *1*) *else* (*length A* + *1*))
length (*A* ∨∗ *B*) = (*if* (*length A* ≤ *length B*) *then* (*length B* + *1*) *else* (*length A* + *1*))
length (*ff*) = *0*

abbreviation
less-prod-nat (- <∗ - [*50*,*50*]*50*) **where**
p <∗ *q* ≡ (*p*,*q*) : *less-than* <∗*lex*∗> *less-than*

lemma *nat-prod-induct* [*case-names less*]:
fixes *x y* :: *nat*
assumes *induct-step*: ⋀ *x y*. (⋀ *u v*. (*u*, *v*) <∗ (*x*, *y*) ⟹ *P u v*) ⟹ *P x y*
shows *P x y*
proof −
have *wf* (*less-than* <∗*lex*∗> *less-than*) **by** *blast*
then show *?thesis*
proof (*induct p* ≡ (*x*, *y*) *arbitrary*: *x y*)
case (*less p*)
show *P x y*
proof (*rule induct-step*)
fix *u v*
assume (*u*, *v*) <∗ (*x*, *y*)
with *less* **show** *P u v* **by** *simp*
qed
qed
qed

lemma *midMultiset*:
assumes Γ ⊕ *A* = Γ′ ⊕ *B* **and** *A* ≠ *B*
shows ∃ Γ″. Γ = Γ″ ⊕ *B* ∧ Γ′ = Γ″ ⊕ *A*
proof−
from *assms* **have** *A* :# Γ′
proof−
from *assms* **have** *set-of* (Γ ⊕ *A*) = *set-of* (Γ′ ⊕ *B*) **by** *auto*

```
    then have set-of Γ ∪ {A} = set-of Γ' ∪ {B} by auto
    then have set-of Γ ∪ {A} ⊆ set-of Γ' ∪ {B} by simp
    then have A ∈ set-of Γ' using assms by auto
    thus A :# Γ' by simp
    qed
  then have Γ' ⊖ A ⊕ A = Γ' by (auto simp add:multiset-eq-conv-count-eq)
  then have ∃ Γ''. Γ' = Γ'' ⊕ A apply (rule-tac x=Γ' ⊖ A in exI) by auto
  then obtain Γ'' where eq1:Γ' = Γ'' ⊕ A by blast
  from ‹Γ ⊕ A = Γ' ⊕ B› eq1 have Γ ⊕ A = Γ'' ⊕ A ⊕ B by auto
  then have Γ = Γ'' ⊕ B by (auto simp add:multiset-eq-conv-count-eq)
  thus ?thesis using eq1 by blast
qed

lemma inversionImpL:
  assumes Γ ⊕ A⊃B ⇒ C ↓ n
  shows ∃ j. j≤n ∧ Γ ⊕ B ⇒ C ↓ j
  using assms
proof (induct Γ≡Γ ⊕ A⊃B C n arbitrary:Γ)
  case (Ax i Γ')
  then have Atom i :# Γ by auto
  then have Γ⊕B⇒ Atom i ↓ 0 by auto
  then show ?case by blast
next
  case (LBot Γ' C)
  then have ff :# Γ by auto
  then have Γ ⊕ B⇒ C ↓ 0 by auto
  then show ?case by blast
next
  case (ImpR Γ' E F k)
  then have Γ' ⊕ E = Γ ⊕ A⊃B ⊕ E by auto
  then have ∃ j. j≤k ∧ Γ ⊕ B ⊕ E ⇒ F ↓ j using prems(3)[where Γ=Γ⊕E] by (auto simp
add:union-ac)
  then obtain j where c1: j≤k
                and c2: Γ ⊕ B ⊕ E ⇒ F ↓ j by auto
  from c2 have Γ ⊕ B ⇒ E⊃F ↓ j+1 using provable-dp.ImpR[where Γ=Γ ⊕ B and A=E and
B=F] by auto
  then show ?case using c1 by auto
next
  case (ConjR Γ' E k F l)
  then have ∃j≤k. Γ ⊕ B ⇒ E ↓ j and ∃j≤l. Γ ⊕ B ⇒ F ↓ j by auto
  then obtain j1 j2 where c1: j1 ≤ k
                      and c2: Γ ⊕ B ⇒ E ↓ j1
                      and c3: j2 ≤ l
```

```
                and c4: Γ ⊕ B ⇒ F ↓ j2 by auto
  then show ?case using provable-dp.ConjR[where Γ=Γ⊕B and n=j1 and m=j2 and A=E
and B=F]
    apply (rule-tac x=j1+j2+1 in exI) by auto
next
  case (ConjL Γ' E F C n Γ'')
  then obtain Γ1 where eq1: Γ' = Γ1 ⊕ A⊃B
                and eq2: Γ'' = Γ1 ⊕ E∧*F using midMultiset[where Γ=Γ' and A=E∧*F and
Γ'=Γ'' and B=A⊃B] by auto
  from eq1 prems(3)[where Γ=Γ1⊕E⊕F] have ∃ j≤n. Γ1 ⊕ E ⊕ F ⊕ B ⇒ C ↓ j by (auto
simp add:union-ac)
  then obtain j where eq3: j≤n and Γ1 ⊕ E ⊕ F ⊕ B ⇒ C ↓ j by blast
  then have Γ1 ⊕ E∧*F ⊕ B ⇒ C ↓ j+1 using provable-dp.ConjL[where Γ=Γ1⊕B and A=E
and B=F] by (auto simp add:union-ac)
  then show ?case using eq2 eq3 by auto
next
  case (DisjR1 Γ' E n F)
  then have ∃j≤n. Γ ⊕ B ⇒ E ↓ j by auto
  then obtain j where eq:j≤n and Γ ⊕ B ⇒ E ↓ j by blast
  then have Γ ⊕ B ⇒ E∨*F ↓ j+1 using provable-dp.DisjR1 by auto
  then show ?case using eq by auto
next
  case (DisjR2 Γ' F n E)
  then have ∃j≤n. Γ ⊕ B ⇒ F ↓ j by auto
  then obtain j where eq:j≤n and Γ ⊕ B ⇒ F ↓ j by blast
  then have Γ ⊕ B ⇒ E∨*F ↓ j+1 using provable-dp.DisjR2 by auto
  then show ?case using eq by auto
next
  case (DisjL Γ' E C n F m Γ'')
  then obtain Γ1 where eq1: Γ' = Γ1 ⊕ A⊃B
                and eq2: Γ'' = Γ1 ⊕ E∨*F using midMultiset[where Γ=Γ' and Γ'=Γ'' and
A=E∨*F and B=A⊃B] by auto
  from eq1 prems(3)[where Γ=Γ1⊕E] have ∃ j≤n. Γ1 ⊕ E ⊕ B ⇒ C ↓ j by (auto simp
add:union-ac)
  moreover
  from eq1 prems(5)[where Γ=Γ1⊕F] have ∃ k≤m. Γ1 ⊕ F ⊕ B ⇒ C ↓ k by (auto simp
add:union-ac)
  ultimately
  obtain j k where a: j≤n ∧ k≤m
            and b: Γ1 ⊕ E ⊕ B ⇒ C ↓ j
            and c: Γ1 ⊕ F ⊕ B ⇒ C ↓ k by blast
  from b c have Γ1 ⊕ E∨*F ⊕ B ⇒ C ↓ j+k+1 using provable-dp.DisjL[where Γ=Γ1⊕B and
A=E and B=F]
```

```
    by (auto simp add:union-ac)
  then show ?case using a eq2 apply (rule-tac x=j+k+1 in exI) by auto
next
  case (ImpL Γ' E F n C m Γ'')
  have E ⊃ F = A ⊃ B ∨ E ⊃ F ≠ A ⊃ B by blast
    moreover
    {assume E ⊃ F = A ⊃ B
     then have Γ' = Γ'' using prems by auto
     then have Γ'' ⊕ B ⇒ C ↓ m using prems by auto
     then have ∃ k. k≤n+m+1 ∧ Γ'' ⊕ B ⇒ C ↓ k apply (rule-tac x=m in exI) by auto
    }
    moreover
    {assume a:E ⊃ F ≠ A ⊃ B
     from prems obtain Γ1 where eq1: Γ' = Γ1 ⊕ A⊃B
                          and eq2: Γ'' = Γ1 ⊕ E⊃F using midMultiset[where Γ=Γ' and Γ'=Γ''
and A=E⊃F and B=A⊃B]
       by auto
    from prems have ∃ j. j≤n ∧ Γ'' ⊕ B ⇒ E ↓ j by auto
    then obtain j where b1: j≤n
                  and b2: Γ1⊕ B ⊕ E⊃F ⇒ E ↓ j using eq2 by (auto simp add:union-ac)
    moreover
    from eq1 have ∃ k. k≤m ∧ Γ1 ⊕ F ⊕ B ⇒ C ↓ k using
          prems(5)[where Γ=Γ1 ⊕ F] by (auto simp add:union-ac)
    then obtain k where c1: k≤m
                  and c2: Γ1 ⊕ F ⊕ B ⇒ C ↓ k by auto
    ultimately
     have Γ1 ⊕ B ⊕ E⊃F ⇒ C ↓ j+k+1 using provable-dp.ImpL[where Γ=Γ1⊕B and A=E
and B=F]
       by (auto simp add:union-ac)
    then have ∃ k. k≤n+m+1 ∧ Γ'' ⊕ B ⇒ C ↓ k using b1 c1 eq2 apply (rule-tac x=j+k+1
in exI)
       by (auto simp add:union-ac)
    }
    ultimately
    show ?case by blast
qed

lemma inversionConjL:
  assumes Γ ⊕ A∧*B ⇒C ↓ n
  shows ∃ j. j≤n ∧ Γ ⊕ A ⊕ B ⇒ C ↓ j
  using assms
proof (induct Γ≡Γ ⊕ A∧*B C n arbitrary: Γ)
  case (Ax i Γ')
```

```
  then have Atom i :# Γ by auto
  then have Γ ⊕ A ⊕ B ⇒ Atom i ↓ 0 by auto
  then show ?case by blast
next
  case (LBot Γ' C)
  then have ff :# Γ by auto
  then have Γ ⊕ A ⊕ B ⇒ C ↓ 0 by auto
  then show ?case by blast
next
  case (ImpR Γ' E F k)
  then have Γ' ⊕ E = Γ ⊕ A∧*B ⊕ E by auto
  then have ∃ j. j≤k ∧ Γ ⊕ A ⊕ B ⊕ E ⇒ F ↓ j using prems(3)[where Γ=Γ⊕E] by (auto
simp add:union-ac)
  then obtain j where c1: j≤k
                  and c2: Γ ⊕ A ⊕ B ⊕ E ⇒ F ↓ j by auto
  from c2 have Γ ⊕ A ⊕ B ⇒ E⊃F ↓ j+1 using provable-dp.ImpR[where Γ=Γ ⊕ A ⊕ B and
A=E and B=F] by auto
  then show ?case using c1 by auto
next
  case (ConjR Γ' E k F l)
  then have ∃j≤k. Γ ⊕ A ⊕ B ⇒ E ↓ j and ∃j≤l. Γ ⊕ A ⊕  B ⇒ F ↓ j by auto
  then obtain j1 j2 where c1: j1 ≤ k
                      and c2: Γ ⊕ A ⊕ B ⇒ E ↓ j1
                      and c3: j2 ≤ l
                      and c4: Γ ⊕ A ⊕ B ⇒ F ↓ j2 by auto
  then show ?case using provable-dp.ConjR[where Γ=Γ⊕A⊕B and n=j1 and m=j2 and A=E
and B=F]
    apply (rule-tac x=j1+j2+1 in exI) by auto
next
  case (DisjR1 Γ' E n F)
  then have ∃j≤n. Γ ⊕ A ⊕ B ⇒ E ↓ j by auto
  then obtain j where eq:j≤n and Γ ⊕ A ⊕ B ⇒ E ↓ j by blast
  then have Γ ⊕ A ⊕ B ⇒ E∨*F ↓ j+1 using provable-dp.DisjR1 by auto
  then show ?case using eq by auto
next
  case (DisjR2 Γ' F n E)
  then have ∃j≤n. Γ ⊕ A ⊕ B ⇒ F ↓ j by auto
  then obtain j where eq:j≤n and Γ ⊕ A ⊕ B ⇒ F ↓ j by blast
  then have Γ ⊕ A ⊕ B ⇒ E∨*F ↓ j+1 using provable-dp.DisjR2 by auto
  then show ?case using eq by auto
next
  case (DisjL Γ' E C n F m Γ'')
  then obtain Γ1 where eq1: Γ' = Γ1 ⊕ A∧*B
```

and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \vee * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = E \vee * F$ **and** $B = A \wedge * B$] **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus A \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
moreover
from *eq1 prems*(5)[**where** $\Gamma = \Gamma 1 \oplus F$] **have** $\exists\ k \leq m.\ \Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
ultimately
obtain $j\ k$ **where** *a*: $j \leq n \wedge k \leq m$
and *b*: $\Gamma 1 \oplus E \oplus A \oplus B \Rightarrow C \downarrow j$
and *c*: $\Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b c* **have** $\Gamma 1 \oplus E \vee * F \oplus A \oplus B \Rightarrow C \downarrow j+k+1$ **using** *provable-dp.DisjL*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a eq2* **apply** (*rule-tac* $x = j+k+1$ **in** *exI*) **by** *auto*
next
case (*ImpL* $\Gamma'\ E\ F\ n\ C\ m\ \Gamma''$)
from *prems* **obtain** $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \wedge * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \supset F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = E \supset F$ **and** $B = A \wedge * B$]
by *auto*
from *prems* **have** $\exists\ j.\ j \leq n \wedge \Gamma'' \oplus A \oplus B \Rightarrow E \downarrow j$ **by** *auto*
then obtain j **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus A \oplus B \oplus E \supset F \Rightarrow E \downarrow j$ **using** *eq2* **by** (*auto simp add*:*union-ac*)
moreover
from *eq1* **have** $\exists\ k.\ k \leq m \wedge \Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **using**
prems(5)[**where** $\Gamma = \Gamma 1 \oplus F$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *c1*: $k \leq m$
and *c2*: $\Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **by** *auto*
ultimately
have $\Gamma 1 \oplus A \oplus B \oplus E \supset F \Rightarrow C \downarrow j+k+1$ **using** *provable-dp.ImpL*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
then have $\exists\ k.\ k \leq n+m+1 \wedge \Gamma'' \oplus A \oplus B \Rightarrow C \downarrow k$ **using** *b1 c1 eq2* **apply** (*rule-tac* $x = j+k+1$ **in** *exI*)
by (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*ConjL* $\Gamma'\ E\ F\ C\ n\ \Gamma''$)
have $E \wedge * F = A \wedge * B \vee E \wedge * F \neq A \wedge * B$ **by** *blast*
moreover
{assume $E \wedge * F = A \wedge * B$
then have $\Gamma' = \Gamma'' \wedge E = A \wedge F = B$ **using** *prems* **by** *auto*

then have $\exists\ j.\ j{\leq}n{+}1 \wedge \Gamma'' \oplus A \oplus B \Rightarrow C \downarrow j$ **using** *prems* **apply** (*rule-tac x=n* **in** *exI*) **by** *auto*
}
moreover
{assume $E{\wedge}{*}F \neq A{\wedge}{*}B$
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A{\wedge}{*}B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E{\wedge}{*}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}E{\wedge}{*}F$ **and** $B{=}A{\wedge}{*}B$] *prems*
by *auto*
from *prems* **have** $\exists\ j.\ j{\leq}n \wedge \Gamma 1 \oplus A \oplus B \oplus E \oplus F \Rightarrow C \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\oplus}F$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus A \oplus B \oplus E \oplus F \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
from *b2* **have** $\Gamma 1 \oplus A \oplus B \oplus E{\wedge}{*}F \Rightarrow C \downarrow j{+}1$ **using** *provable-dp.ConjL*[**where** $\Gamma{=}\Gamma 1{\oplus}A{\oplus}B$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ j{\leq}n{+}1.\ \Gamma'' \oplus A \oplus B \Rightarrow C \downarrow j$ **using** *eq2 b1* **apply** (*rule-tac x=j+1* **in** *exI*) **by** (*auto simp add*:*union-ac*)
}
ultimately
show *?case* **by** *blast*
qed

lemma *inversionDisjL*:
assumes $\Gamma \oplus A{\vee}{*}B \Rightarrow C \downarrow n$
shows $\exists\ j\ k.\ j{\leq}n \wedge k{\leq}n \wedge \Gamma \oplus A \Rightarrow C \downarrow j \wedge \Gamma \oplus B \Rightarrow C \downarrow k$
using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma{\oplus}A{\vee}{*}B$ *C n arbitrary*: Γ)
case (*Ax i* Γ')
then have *Atom i* :# Γ **by** *auto*
then have $\Gamma \oplus A \Rightarrow$ *Atom i* $\downarrow$ *0* **and** $\Gamma \oplus B \Rightarrow$ *Atom i* $\downarrow$ *0* **by** *auto*
then show *?case* **by** *blast*
next
case (*LBot* Γ' *C*)
then have *ff* :# Γ **by** *auto*
then have $\Gamma \oplus A \Rightarrow C \downarrow 0$ **and** $\Gamma \oplus B \Rightarrow C \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*ConjR* Γ' *E k F l*)
then have $\exists\ j1\ j2.\ j1{\leq}k \wedge j2{\leq}k \wedge \Gamma \oplus A \Rightarrow E \downarrow j1 \wedge \Gamma \oplus B \Rightarrow E \downarrow j2$
and $\exists\ j3\ j4.\ j3{\leq}l \wedge j4{\leq}l \wedge \Gamma \oplus A \Rightarrow F \downarrow j3 \wedge \Gamma \oplus B \Rightarrow F \downarrow j4$ **by** *auto*
then obtain *j1 j2 j3 j4* **where** *c*: $j1 \leq k \wedge j2{\leq}k \wedge j3{\leq}l \wedge j4{\leq}l$
and *c1*: $\Gamma \oplus A \Rightarrow E \downarrow j1$

and *c2*: $\Gamma \oplus B \Rightarrow E \downarrow j2$
and *c3*: $\Gamma \oplus A \Rightarrow F \downarrow j3$
and *c4*: $\Gamma \oplus B \Rightarrow F \downarrow j4$ **by** *auto*
from *c1 c3* **have** $\Gamma \oplus A \Rightarrow E \wedge * F \downarrow j1+j3+1$ **using** *provable-dp.ConjR*[**where** $\Gamma=\Gamma \oplus A$] **by** *auto*
moreover
from *c2 c4* **have** $\Gamma \oplus B \Rightarrow E \wedge * F \downarrow j2+j4+1$ **using** *provable-dp.ConjR*[**where** $\Gamma=\Gamma \oplus B$] **by** *auto*
ultimately
show *?case* **using** *c* **apply** (*rule-tac x=j1+j3+1* **in** *exI*, *rule-tac x=j2+j4+1* **in** *exI*) **by** *auto*
next
case (*DisjR1* Γ' *E n F*)
then have $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma \oplus A \Rightarrow E \downarrow j \wedge \Gamma \oplus B \Rightarrow E \downarrow k$ **by** *auto*
then obtain *j k* **where** *eq*:$j \leq n \wedge k \leq n$ **and** $\Gamma \oplus A \Rightarrow E \downarrow j \wedge \Gamma \oplus B \Rightarrow E \downarrow k$ **by** *blast*
then have $\Gamma \oplus A \Rightarrow E \vee * F \downarrow j+1 \wedge \Gamma \oplus B \Rightarrow E \vee * F \downarrow k+1$ **using** *provable-dp.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ' *F n E*)
then have $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma \oplus A \Rightarrow F \downarrow j \wedge \Gamma \oplus B \Rightarrow F \downarrow k$ **by** *auto*
then obtain *j k* **where** *eq*:$j \leq n \wedge k \leq n$ **and** $\Gamma \oplus A \Rightarrow F \downarrow j \wedge \Gamma \oplus B \Rightarrow F \downarrow k$ **by** *blast*
then have $\Gamma \oplus A \Rightarrow E \vee * F \downarrow j+1 \wedge \Gamma \oplus B \Rightarrow E \vee * F \downarrow k+1$ **using** *provable-dp.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*ImpR* Γ' *E F k*)
then have $\Gamma' \oplus E = \Gamma \oplus A \vee * B \oplus E$ **by** *auto*
then have $\exists\ j1\ j2.\ j1 \leq k \wedge j2 \leq k \wedge \Gamma \oplus A \oplus E \Rightarrow F \downarrow j1 \wedge \Gamma \oplus B \oplus E \Rightarrow F \downarrow j2$
using *prems*(*3*)[**where** $\Gamma=\Gamma \oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain *j1 j2* **where** *c1*: $j1 \leq k \wedge j2 \leq k$
and *c2*: $\Gamma \oplus A \oplus E \Rightarrow F \downarrow j1$
and *c3*: $\Gamma \oplus B \oplus E \Rightarrow F \downarrow j2$ **by** *auto*
from *c2* **have** $\Gamma \oplus A \Rightarrow E \supset F \downarrow j1+1$ **using** *provable-dp.ImpR*[**where** $\Gamma=\Gamma \oplus A$ **and** $A=E$ **and** $B=F$] **by** *auto*
moreover
from *c3* **have** $\Gamma \oplus B \Rightarrow E \supset F \downarrow j2+1$ **using** *provable-dp.ImpR*[**where** $\Gamma=\Gamma \oplus B$ **and** $A=E$ **and** $B=F$] **by** *auto*
ultimately
show *?case* **using** *c1* **apply** (*rule-tac x=j1+1* **in** *exI*, *rule-tac x=j2+1* **in** *exI*) **by** *auto*
next
case (*ImpL* Γ' *E F n C m* Γ'')
from *prems* **obtain** $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \vee * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \supset F$ **using** *midMultiset*[**where** $\Gamma=\Gamma'$ **and** $\Gamma'=\Gamma''$ **and** $A=E \supset F$ **and** $B=A \vee * B$]
by *auto*

from *prems* **have** $\exists\ j\ k.\ j{\leq}n \wedge k{\leq}n \wedge \Gamma'' \oplus A \Rightarrow E \downarrow j \wedge \Gamma'' \oplus B \Rightarrow E \downarrow k$ **by** *auto*
then obtain *j k* **where** *b1*: $j{\leq}n \wedge k{\leq}n$
and *b2*: $\Gamma 1 \oplus A \oplus E{\supset}F \Rightarrow E \downarrow j$
and *b3*: $\Gamma 1 \oplus B \oplus E{\supset}F \Rightarrow E \downarrow k$ **using** *eq2* **by** (*auto simp add*:*union-ac*)
from *eq1* **have** $\exists\ j1\ k1.\ j1{\leq}m \wedge k1{\leq}m \wedge \Gamma 1 \oplus F \oplus A \Rightarrow C \downarrow j1 \wedge \Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k1$ **using**
prems(*5*)[**where** $\Gamma{=}\Gamma 1 \oplus F$] **by** (*auto simp add*:*union-ac*)
then obtain *j1 k1* **where** *c1*: $j1{\leq}m \wedge k1{\leq}m$
and *c2*: $\Gamma 1 \oplus F \oplus A \Rightarrow C \downarrow j1$
and *c3*: $\Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k1$ **by** *auto*
from *b2 c2* **have** $\Gamma 1 \oplus A \oplus E{\supset}F \Rightarrow C \downarrow j{+}j1{+}1$ **using** *provable-dp.ImpL*[**where** $\Gamma{=}\Gamma 1{\oplus}A$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
moreover
from *b3 c3* **have** $\Gamma 1 \oplus B \oplus E{\supset}F \Rightarrow C \downarrow k{+}k1{+}1$ **using** *provable-dp.ImpL*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
ultimately
show *?case* **using** *b1 c1 eq2* **apply** (*rule-tac* $x{=}j{+}j1{+}1$ **in** *exI*,*rule-tac* $x{=}k{+}k1{+}1$ **in** *exI*) **by** (*auto simp add*:*union-ac*)
next
case (*ConjL* Γ' *E F C n* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A{\vee}{*}B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E{\wedge}{*}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $A{=}E{\wedge}{*}F$ **and** $\Gamma'{=}\Gamma''$ **and** $B{=}A{\vee}{*}B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\oplus}F$]
have $\exists\ j\ k.\ j{\leq}n \wedge k{\leq}n \wedge \Gamma 1 \oplus E \oplus F \oplus A \Rightarrow C \downarrow j \wedge \Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
then obtain *j k* **where** *eq3*: $j{\leq}n \wedge k{\leq}n$
and *c1*: $\Gamma 1 \oplus E \oplus F \oplus A \Rightarrow C \downarrow j$
and *c2*: $\Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *c1* **have** $\Gamma 1 \oplus E{\wedge}{*}F \oplus A \Rightarrow C \downarrow j{+}1$ **using** *provable-dp.ConjL*[**where** $\Gamma{=}\Gamma 1{\oplus}A$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
moreover
from *c2* **have** $\Gamma 1 \oplus E{\wedge}{*}F \oplus B \Rightarrow C \downarrow k{+}1$ **using** *provable-dp.ConjL*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
ultimately
show *?case* **using** *eq2 eq3* **apply** (*rule-tac* $x{=}j{+}1$ **in** *exI*, *rule-tac* $x{=}k{+}1$ **in** *exI*) **by** *auto*
next
case (*DisjL* Γ' *E C n F m* Γ'')
have $E{\vee}{*}F = A{\vee}{*}B \vee E{\vee}{*}F \neq A{\vee}{*}B$ **by** *blast*

moreover
{assume $E \vee * F = A \vee * B$
from *prems* **have** $\Gamma' = \Gamma'' \wedge E{=}A \wedge F{=}B$ **by** *auto*
then have $\Gamma'' \oplus A \Rightarrow C \downarrow n$ **and** $\Gamma'' \oplus B \Rightarrow C \downarrow m$ **using** *prems* **by** *auto*
then have $\exists\ j\ k.\ j{\leq}n{+}m{+}1 \wedge k{\leq}n{+}m{+}1 \wedge \Gamma'' \oplus A \Rightarrow C \downarrow j \wedge \Gamma'' \oplus B \Rightarrow C \downarrow k$ **apply** (*rule-tac* $x{=}n$ **in** *exI*,*rule-tac* $x{=}m$ **in** *exI*)
by *auto*
}
moreover
{assume $E \vee * F \neq A \vee * B$
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \vee * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \vee * F$
using *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}E \vee * F$ **and** $B{=}A \vee * B$] *prems* **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma{=}\Gamma 1 \oplus E$] **have** $\exists\ j1\ k1.\ j1{\leq}n \wedge k1{\leq}n \wedge \Gamma 1 \oplus E \oplus A \Rightarrow C \downarrow j1 \wedge \Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow k1$
by (*auto simp add*:*union-ac*)
then obtain *j1 k1* **where** *a*: $j1{\leq}n \wedge k1{\leq}n$
and *a1*: $\Gamma 1 \oplus E \oplus A \Rightarrow C \downarrow j1$
and *a2*: $\Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow k1$ **by** *blast*
from *eq1 prems*(5)[**where** $\Gamma{=}\Gamma 1 \oplus F$] **have** $\exists\ j2\ k2.\ j2{\leq}m \wedge k2{\leq}m \wedge \Gamma 1 \oplus F \oplus A \Rightarrow C \downarrow j2 \wedge \Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k2$
by (*auto simp add*:*union-ac*)
then obtain *j2 k2* **where** *b*: $j2{\leq}m \wedge k2{\leq}m$
and *b1*: $\Gamma 1 \oplus F \oplus A \Rightarrow C \downarrow j2$
and *b2*: $\Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k2$ **by** *blast*
from *a1 b1* **have** $\Gamma 1 \oplus E \vee * F \oplus A \Rightarrow C \downarrow j1{+}j2{+}1$ **using** *provable-dp.DisjL*[**where** $\Gamma{=}\Gamma 1 \oplus A$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
moreover
from *a2 b2* **have** $\Gamma 1 \oplus E \vee * F \oplus B \Rightarrow C \downarrow k1{+}k2{+}1$ **using** *provable-dp.DisjL*[**where** $\Gamma{=}\Gamma 1 \oplus B$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
ultimately
have $\exists\ j\ k.\ j{\leq}n{+}m{+}1 \wedge k{\leq}n{+}m{+}1 \wedge \Gamma'' \oplus A \Rightarrow C \downarrow j \wedge \Gamma'' \oplus B \Rightarrow C \downarrow k$ **using** *eq2 a b*
apply (*rule-tac* $x{=}j1{+}j2{+}1$ **in** *exI*, *rule-tac* $x{=}k1{+}k2{+}1$ **in** *exI*) **by** *auto*
}
ultimately
show *?case* **by** *blast*
qed

lemma *dpWeak*:
fixes Γ :: *form multiset*
assumes *a*: $\Gamma \Rightarrow C \downarrow n$

shows $\Gamma \oplus A \Rightarrow C \downarrow n$
using *a*
proof (*induct* Γ C n)
case (*Ax* *i* Γ)
then have *finite* (*set-of* Γ) *Atom* *i* :# Γ **by** *simp-all*
then show $\Gamma \oplus A \Rightarrow$ *Atom* *i* $\downarrow$ *0* **using** *provable-dp.Ax* **by** *auto*
next
case (*LBot* Γ C)
then have *finite* (*set-of* Γ) *ff* :# Γ **by** *simp-all*
then show $\Gamma \oplus A \Rightarrow C \downarrow$ *0* **using** *provable-dp.LBot* **by** *auto*
next
case (*ConjR* Γ C n D m)
then have $\Gamma \oplus A \Rightarrow C \wedge * D \downarrow n+m+1$ **using** *provable-dp.ConjR*[**where** $\Gamma=\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR1* Γ C n D)
then have $\Gamma \oplus A \Rightarrow C \vee * D \downarrow n+1$ **using** *provable-dp.DisjR1*[**where** $\Gamma=\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR2* Γ D n C)
then have $\Gamma \oplus A \Rightarrow C \vee * D \downarrow n+1$ **using** *provable-dp.DisjR2*[**where** $\Gamma=\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*ImpR* Γ C D m)
then have $\Gamma \oplus C \oplus A \Rightarrow D \downarrow m$ **by** *auto*
then show $\Gamma \oplus A \Rightarrow C \supset D \downarrow m+1$ **using** *provable-dp.ImpR* **by** (*auto simp add*:*union-ac*)
next
case (*ConjL* Γ C D E n)
then have $\Gamma \oplus C \wedge * D \oplus A \Rightarrow E \downarrow n+1$ **using** *provable-dp.ConjL*[**where** $\Gamma=\Gamma \oplus A$ **and** $A=C$ **and** $B=D$] **by** (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*DisjL* Γ C E n D m)
then have $\Gamma \oplus C \vee * D \oplus A \Rightarrow E \downarrow n+m+1$ **using** *provable-dp.DisjL*[**where** $\Gamma=\Gamma \oplus A$ **and** $A=C$ **and** $B=D$] **by** (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*ImpL* Γ C D n F m)
then have $\Gamma \oplus C \supset D \oplus A \Rightarrow C \downarrow n$ $\Gamma \oplus D \oplus A \Rightarrow F \downarrow m$**by** *simp*
then show $\Gamma \oplus C \supset D \oplus A \Rightarrow F \downarrow n+m+1$
using *provable-dp.ImpL*[**where** $A=C$ **and** $B=D$ **and** $\Gamma=\Gamma \oplus A$] **by** (*auto simp add*:*union-ac*)
qed

```
lemma dpWeak':
  assumes Γ ⇒ C ↓ n
  shows Γ + Γ' ⇒ C ↓ n
using assms
proof (induct Γ')
  case empty
  then show ?case by auto
next
  case (add Γ' x)
  then have Γ + Γ' ⇒ C ↓ n by auto
  then have Γ + Γ' ⊕ x ⇒ C ↓ n using dpWeak[where Γ=Γ+Γ' and A=x] by (auto simp
only:union-ac)
  then show ?case by blast
qed

lemma cutAdmissibility:
fixes A :: form
  and n m :: nat
assumes Γ ⇒ A ↓ n Γ ⊕ A ⇒ C ↓ m
shows ∃ k. Γ ⇒ C ↓ k
using assms
proof (induct x ≡ length A y ≡ n+m+1  arbitrary: Γ A C n m rule: nat-prod-induct)
   case (less x y)
   then have IH: ⋀ u v Γ A n C m.
                 ⟦(u, v) <* (x, y); Γ ⇒ A ↓ n; Γ ⊕ A ⇒ C ↓ m; u = length A; v = n + m + 1⟧
                       ⟹ ∃ a. Γ ⇒ C ↓ a by auto
   have Γ ⇒ A ↓ n by fact
   then show ?case
       proof (cases)
       case (Ax i Γ')
       with ‹Γ ⊕ A ⇒ C ↓ m› have Γ ⊕ Atom i ⇒ C ↓ m by simp
       then show ∃ k. Γ ⇒ C ↓ k
           proof (cases)
           case (Ax j Γ1)
           have j = i ∨ j ≠ i by blast
           moreover
             {assume j=i
              then have Γ ⇒ C ↓ 0 using ‹C = Atom j› and ‹Γ ⇒ A ↓ n› and ‹A = Atom i›
and ‹n=0› by auto
              then have ∃ n. Γ ⇒ C ↓ n by blast
             }
           moreover
             {assume j ≠ i
```

```
            then have Atom j :# Γ using ⟨Atom j :# Γ1⟩ and ⟨Γ ⊕ Atom i = Γ1⟩ by auto
            then have Γ ⇒ C ↓ 0 using ⟨C = Atom j⟩ by auto
            then have ∃ n. Γ ⇒ C ↓ n by blast
           }
         ultimately show ∃ n. Γ ⇒ C ↓ n by auto
      next
         case (LBot Γ1 C')
         then have ff :# Γ by auto
         then have Γ ⇒ C ↓ 0 by auto
         then show ∃ n. Γ ⇒ C ↓ n by auto
      next
         case (ImpR Γ1 E F j)
         from ⟨A = Atom i⟩ and ⟨n=0⟩ and ⟨Γ ⇒ A ↓ n⟩ have Γ ⇒ Atom i ↓ 0 by simp
         then have Γ ⊕ E ⇒ Atom i ↓ 0 using dpWeak by auto
         moreover from ⟨Γ1 ⊕ E ⇒ F ↓ j⟩ and ⟨Γ ⊕ Atom i = Γ1⟩
            have Γ ⊕ E ⊕ Atom i ⇒ F ↓ j by (auto simp add:union-ac)
        ultimately have ∃ n. Γ ⊕ E ⇒ F ↓ n using IH[where A=Atom i and Γ=Γ⊕E and
n=0 and m=j and C=F]
           and ⟨m = j+1⟩ and ⟨x = length A⟩ and ⟨y = n+m+1⟩ and ⟨A = Atom i⟩ by auto
         then have ∃ n. Γ ⇒ E⊃F ↓ n using provable-dp.ImpR by auto
         then show ∃ n. Γ ⇒ C ↓ n using ⟨C = E ⊃ F⟩ by simp
      next
         case (ImpL Γ1 E F j C' k)
         then have Γ ⊕ Atom i ⇒ E ↓ j by simp
        moreover from ⟨A = Atom i⟩ and ⟨n=0⟩ and ⟨Γ ⇒ A ↓ n⟩ have Γ ⇒ Atom i ↓ 0 by
simp
         ultimately have ∃ n. Γ ⇒ E ↓ n using IH[where Γ=Γ and A=Atom i and n=0
and C=E and m=j]
           and ⟨A = Atom i⟩ and ⟨y = n+m+1⟩ and ⟨m=j+k+1⟩ by auto
         from ⟨Γ ⊕ Atom i = Γ1 ⊕ E⊃F⟩ obtain Γ2 where
           Γ = Γ2 ⊕ E⊃F and
           Γ1 = Γ2 ⊕ Atom i using midMultiset[where A=Atom i and B=E⊃F] by auto
         with ⟨Atom i :# Γ'⟩ and ⟨Γ = Γ'⟩ have Atom i :# Γ2 by auto
         then have Γ2 ⊕ F ⇒ Atom i ↓ 0 by auto
        moreover from ⟨Γ1 ⊕ F ⇒ C' ↓ k⟩ and ⟨Γ1 = Γ2 ⊕ Atom i⟩ have Γ2 ⊕ F ⊕ Atom
i ⇒ C' ↓ k
           by (auto simp add:union-ac)
         ultimately have ∃ n. Γ2 ⊕ F ⇒ C' ↓ n using IH[where Γ=Γ2⊕F and A=Atom i
and C=C' and n=0 and m=k]
           and ⟨A = Atom i⟩ and ⟨y=n+m+1⟩ and ⟨m=j+k+1⟩ by auto
          with ⟨∃ n. Γ ⇒ E ↓ n⟩ and ⟨Γ = Γ2 ⊕ E⊃F⟩ have ∃ n. Γ ⇒ C' ↓ n using
provable-dp.ImpL by auto
         with ⟨C = C'⟩ show ∃ n. Γ ⇒ C ↓ n by auto
```

next
case (*ConjR* Γ*1 E j F k*)
with ‹*A* = *Atom i*› **and** ‹Γ ⇒ *A* ↓ *n*› **and** *IH*[**where** Γ=Γ **and** *A*=*A* **and** *n*=*n* **and** *C*=*E* **and** *m*=*j*]
and ‹*y*=*n*+*m*+*1*› **have** ∃ *n*. Γ ⇒ *E* ↓ *n* **by** *auto*
moreover from ‹*A* = *Atom i*› **and** ‹Γ ⇒ *A* ↓ *n*› **and** *IH*[**where** Γ=Γ **and** *A*=*A* **and** *n*=*n* **and** *C*=*F* **and** *m*=*k*]
and ‹*y*=*n*+*m*+*1*› **and** ‹Γ*1* ⇒ *F* ↓ *k*› **and** ‹Γ ⊕ *Atom i* = Γ*1*› **and** ‹*m*=*j*+*k*+*1*›
have ∃ *n*. Γ ⇒ *F* ↓ *n* **by** *auto*
ultimately show ∃ *n*. Γ ⇒ *C* ↓ *n* **using** *provable-dp.ConjR* **and** ‹*C* = *E* ∧∗*F*› **by** *auto*
next
case (*DisjR1* Γ*1 E j F*)
with ‹*A* = *Atom i*› **and** ‹Γ ⇒ *A* ↓ *n*› **and** *IH*[**where** Γ=Γ **and** *A*=*A* **and** *n*=*n* **and** *C*=*E* **and** *m*=*j*]
and ‹*y*=*n*+*m*+*1*› **have** ∃ *n*. Γ ⇒ *E* ↓ *n* **by** *auto*
then show ∃ *n*. Γ ⇒ *C* ↓ *n* **using** *provable-dp.DisjR1* **and** ‹*C* = *E*∨∗*F*› **by** *auto*
next
case (*DisjR2* Γ*1 F k E*)
with ‹*A* = *Atom i*› **and** ‹Γ ⇒ *A* ↓ *n*› **and** *IH*[**where** Γ=Γ **and** *A*=*A* **and** *n*=*n* **and** *C*=*F* **and** *m*=*k*]
and ‹*y*=*n*+*m*+*1*› **have** ∃ *n*. Γ ⇒ *F* ↓ *n* **by** *auto*
then show ∃ *n*. Γ ⇒ *C* ↓ *n* **using** *provable-dp.DisjR1* **and** ‹*C* = *E*∨∗*F*› **by** *auto*
next
case (*DisjL* Γ*1 E C′ j F k*)
then obtain Γ*2* **where** Γ = Γ*2* ⊕ *E*∨∗*F*
and Γ*1* = Γ*2* ⊕ *Atom i*
using *midMultiset*[**where** *A*=*Atom i* **and** *B*=*E*∨∗*F*] **by** *auto*
with ‹*Atom i* :# Γ′› **and** ‹Γ = Γ′› **have** *Atom i* :# Γ*2* **by** *auto*
then have Γ*2* ⊕ *E* ⇒ *Atom i* ↓ *0* **and** Γ*2* ⊕ *F* ⇒ *Atom i* ↓ *0* **by** *auto*
from ‹Γ*1* ⊕ *E* ⇒ *C′* ↓ *j*› **and** ‹Γ*1* = Γ*2* ⊕ *Atom i*› **have** Γ*2* ⊕ *E* ⊕ *Atom i* ⇒ *C′* ↓ *j* **by** (*auto simp add*:*union-ac*)
with ‹Γ*2* ⊕ *E* ⇒ *Atom i* ↓ *0*› **and** *IH*[**where** Γ=Γ*2*⊕*E* **and** *A*=*Atom i* **and** *n*=*0* **and** *C*=*C′* **and** *m*=*j*]
and ‹*y* = *n*+*m*+*1*› **and** ‹*m*=*j*+*k*+*1*› **and** ‹*A* = *Atom i*›
have ∃ *n*. Γ*2* ⊕ *E* ⇒ *C′* ↓ *n* **by** *auto*
moreover from ‹Γ*1* ⊕ *F* ⇒ *C′* ↓ *k*› **and** ‹Γ*1* = Γ*2* ⊕ *Atom i*› **have** Γ*2* ⊕ *F* ⊕ *Atom i* ⇒ *C′* ↓ *k* **by** (*auto simp add*:*union-ac*)
with ‹Γ*2* ⊕ *F* ⇒ *Atom i* ↓ *0*› **and** *IH*[**where** Γ=Γ*2*⊕*F* **and** *A*=*Atom i* **and** *n*=*0* **and** *C*=*C′* **and** *m*=*k*]
and ‹*y* = *n*+*m*+*1*› **and** ‹*m*=*j*+*k*+*1*› **and** ‹*A* = *Atom i*›
have ∃ *n*. Γ*2* ⊕ *F* ⇒ *C′* ↓ *n* **by** *auto*
ultimately have ∃ *n*. Γ*2* ⊕ *E*∨∗*F* ⇒ *C* ↓ *n* **using** *provable-dp.DisjL* **and** ‹*C* = *C′*›

by *auto*
then show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **using** $\langle \Gamma = \Gamma 2 \oplus E \vee * F \rangle$ **by** *simp*
next
case (*ConjL* $\Gamma 1\ E\ F\ C'\ j$)
then obtain $\Gamma 2$ **where** $\Gamma = \Gamma 2 \oplus E \wedge * F$
and $\Gamma 1 = \Gamma 2 \oplus Atom\ i$ **using** *midMultiset*[**where** $A{=}Atom\ i$ **and** $B{=}E \wedge * F$] **by** *auto*
with $\langle Atom\ i :\#\ \Gamma' \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **have** $Atom\ i :\#\ \Gamma 2$ **by** *auto*
then have $\Gamma 2 \oplus E \oplus F \Rightarrow Atom\ i \downarrow 0$ **by** *auto*
moreover from $\langle \Gamma 1 \oplus E \oplus F \Rightarrow C' \downarrow j \rangle$ **and** $\langle \Gamma 1 = \Gamma 2 \oplus Atom\ i \rangle$ **have** $\Gamma 2 \oplus E \oplus F \oplus Atom\ i \Rightarrow C' \downarrow j$
by (*auto simp add*:*union-ac*)
ultimately have $\exists\ n.\ \Gamma 2 \oplus E \oplus F \Rightarrow C' \downarrow n$ **using** *IH*[**where** $\Gamma{=}\Gamma 2 \oplus E \oplus F$ **and** $A{=}Atom\ i$ **and** $n{=}0$ **and** $C{=}C'$ **and** $m{=}j$]
and $\langle A = Atom\ i \rangle$ **and** $\langle y = n{+}m{+}1 \rangle$ **and** $\langle m{=}j{+}1 \rangle$ **by** *auto*
then have $\exists\ n.\ \Gamma 2 \oplus E \wedge * F \Rightarrow C \downarrow n$ **using** *provable-dp.ConjL* **and** $\langle C{=}C' \rangle$ **by** *auto*
then show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **using** $\langle \Gamma = \Gamma 2 \oplus E \wedge * F \rangle$ **by** *simp*
qed
next
case (*LBot* $\Gamma'\ A'$)
then have $\Gamma \Rightarrow C \downarrow 0$ **by** *auto*
then show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *blast*
next
case (*ConjR* $\Gamma'\ E\ j\ F\ k$)
with $\langle \Gamma \oplus A \Rightarrow C \downarrow m \rangle$ **have** $\Gamma \oplus E \wedge * F \Rightarrow C \downarrow m$ **by** *simp*
then obtain m' **where** $\Gamma \oplus E \oplus F \Rightarrow C \downarrow m'$ **using** *inversionConjL*[**where** $n{=}m$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*
from $\langle \Gamma' \Rightarrow E \downarrow j \rangle$ **and** $\langle \Gamma{=}\Gamma' \rangle$ **have** $\Gamma \oplus F \Rightarrow E \downarrow j$ **using** *dpWeak* **by** *auto*
with $\langle \Gamma \oplus E \oplus F \Rightarrow C \downarrow m' \rangle$ **and** *IH*[**where** $\Gamma{=}\Gamma \oplus F$ **and** $A{=}E$ **and** $n{=}j$ **and** $m{=}m'$ **and** $C{=}C$]
and $\langle x{=}\ length\ A \rangle$ **and** $\langle A = E \wedge * F \rangle$
have $\exists\ n.\ \Gamma \oplus F \Rightarrow C \downarrow n$ **by** (*auto simp add*:*union-ac*)
with $\langle \Gamma' \Rightarrow F \downarrow k \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** *IH*[**where** $\Gamma{=}\Gamma$ **and** $A{=}F$ **and** $C{=}C$ **and** $n{=}k$]
and $\langle x = length\ A \rangle$ **and** $\langle A = E \wedge * F \rangle$
show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*DisjR1* $\Gamma'\ E\ j\ F$)
with $\langle \Gamma \oplus A \Rightarrow C \downarrow m \rangle$ **have** $\Gamma \oplus E \vee * F \Rightarrow C \downarrow m$ **by** *simp*
then obtain m' **where** $\Gamma \oplus E \Rightarrow C \downarrow m'$ **using** *inversionDisjL*[**where** $n{=}m$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*
with $\langle \Gamma = \Gamma' \rangle$ **and** $\langle \Gamma' \Rightarrow E \downarrow j \rangle$ **and** *IH*[**where** $\Gamma{=}\Gamma$ **and** $A{=}E$ **and** $n{=}j$ **and** $m{=}m'$ **and** $C{=}C$]
and $\langle x = length\ A \rangle$ **and** $\langle A = E \vee * F \rangle$

show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*

next

case (*DisjR2* $\Gamma'\ F\ k\ E$)

with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **have** $\Gamma \oplus E \vee * F \Rightarrow C \downarrow m$ **by** *simp*

then obtain m' **where** $\Gamma \oplus F \Rightarrow C \downarrow m'$ **using** *inversionDisjL*[**where** $n{=}m$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*

with ‹$\Gamma = \Gamma'$› **and** ‹$\Gamma' \Rightarrow F \downarrow k$› **and** *IH*[**where** $\Gamma{=}\Gamma$ **and** $A{=}F$ **and** $n{=}k$ **and** $m{=}m'$ **and** $C{=}C$]

and ‹$x = \mathit{length}\ A$› **and** ‹$A = E \vee * F$›

show $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*

next

case (*ConjL* $\Gamma'\ E\ F\ A'\ j$)

with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **and** ‹$A = A'$› **have** $\Gamma' \oplus E \wedge * F \oplus A' \Rightarrow C \downarrow m$ **by** *simp*

then obtain m' **where** $m' \leq m$ **and** $\Gamma' \oplus E \oplus F \oplus A' \Rightarrow C \downarrow m'$

using *inversionConjL*[**where** $\Gamma{=}\Gamma' \oplus A'$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}C$ **and** $n{=}m$] **by** (*auto simp add*:*union-ac*)

with ‹$\Gamma' \oplus E \oplus F \Rightarrow A' \downarrow j$› **have** $\exists\ n.\ \Gamma' \oplus E \oplus F \Rightarrow C \downarrow n$

using *IH*[**where** $\Gamma{=}\Gamma' \oplus E \oplus F$ **and** $A{=}A'$ **and** $n{=}j$ **and** $m{=}m'$ **and** $C{=}C$]

and ‹$A = A'$› **and** ‹$x = \mathit{length}\ A$› **and** ‹$y = n{+}m{+}1$› **and** ‹$n{=}j{+}1$› **by** *auto*

then have $\exists\ n.\ \Gamma' \oplus E \wedge * F \Rightarrow C \downarrow n$ **using** *provable-dp.ConjL* **by** *auto*

with ‹$\Gamma = \Gamma' \oplus E \wedge * F$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *simp*

next

case (*DisjL* $\Gamma'\ E\ A'\ j\ F\ k$)

with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **have** $\Gamma' \oplus E \vee * F \oplus A' \Rightarrow C \downarrow m$ **by** *simp*

then obtain *m1 m2* **where** $m1 \leq m$ **and** $m2 \leq m$

and $\Gamma' \oplus E \oplus A' \Rightarrow C \downarrow m1$

and $\Gamma' \oplus F \oplus A' \Rightarrow C \downarrow m2$

using *inversionDisjL*[**where** $\Gamma{=}\Gamma' \oplus A'$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}C$ **and** $n{=}m$] **by** (*auto simp add*:*union-ac*)

from ‹$\Gamma' \oplus E \Rightarrow A' \downarrow j$› **and** ‹$\Gamma' \oplus E \oplus A' \Rightarrow C \downarrow m1$›

have $\exists\ n.\ \Gamma' \oplus E \Rightarrow C \downarrow n$

using *IH*[**where** $\Gamma{=}\Gamma' \oplus E$ **and** $A{=}A'$ **and** $n{=}j$ **and** $C{=}C$ **and** $m{=}m1$] **and** ‹$m1 \leq m$›

and ‹$A{=}A'$› **and** ‹$x = \mathit{length}\ A$› **and** ‹$y = n{+}m{+}1$› **and** ‹$n{=}j{+}k{+}1$› **by** *auto*

moreover from ‹$\Gamma' \oplus F \Rightarrow A' \downarrow k$› **and** ‹$\Gamma' \oplus F \oplus A' \Rightarrow C \downarrow m2$›

have $\exists\ n.\ \Gamma' \oplus F \Rightarrow C \downarrow n$

using *IH*[**where** $\Gamma{=}\Gamma' \oplus F$ **and** $A{=}A'$ **and** $n{=}k$ **and** $C{=}C$ **and** $m{=}m2$] **and** ‹$m2 \leq m$›

and ‹$A{=}A'$› **and** ‹$x = \mathit{length}\ A$› **and** ‹$y = n{+}m{+}1$› **and** ‹$n{=}j{+}k{+}1$› **by** *auto*

ultimately have $\exists\ n.\ \Gamma' \oplus E \vee * F \Rightarrow C \downarrow n$ **using** *provable-dp.DisjL* **by** *auto*

with ‹$\Gamma = \Gamma' \oplus E \vee * F$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *simp*

next

case (*ImpL* $\Gamma'\ E\ F\ j\ A'\ k$)

with ‹$\Gamma \oplus A \Rightarrow C \downarrow m$› **and** ‹$A = A'$› **have** $\Gamma' \oplus E \supset F \oplus A' \Rightarrow C \downarrow m$ **by** *simp*

then obtain m' **where** $m' \leq m$

```
                and   Γ' ⊕ F ⊕ A' ⇒ C ↓ m'
        using inversionImpL[where Γ=Γ'⊕A' and A=E and B=F and n=m and C=C] by
(auto simp add:union-ac)
      with ‹Γ' ⊕ F ⇒ A' ↓ k› have ∃ n. Γ'⊕F ⇒ C ↓ n
          using IH[where Γ=Γ'⊕F and A=A' and C=C and n=k and m=m'] and ‹y
=n+m+1› and ‹n=j+k+1›
          and ‹x = length A› and ‹A = A'› by auto
     with ‹Γ' ⊕ E ⊃ F ⇒ E ↓ j› have ∃ n. Γ' ⊕ E⊃F ⇒ C ↓ n using provable-dp.ImpL[where
Γ=Γ' and A=E and B=F]
          by auto
      with ‹Γ = Γ' ⊕ E⊃F› show ∃ n. Γ ⇒ C ↓ n by simp
   next
      case (ImpR Γ' E F j)
      with ‹Γ ⊕ A ⇒ C ↓ m› have Γ' ⊕ E⊃F ⇒ C ↓ m by simp
      then show ∃ n. Γ ⇒ C ↓ n
          proof (cases)
          case (Ax i Γ1)
          then have Atom i :# Γ' by auto
          with ‹Γ = Γ'› have Γ ⇒ Atom i ↓ 0 by auto
          with ‹C = Atom i› show ∃ n. Γ ⇒ C ↓ n by auto
      next
          case (LBot Γ1 C')
          then have ff :# Γ using ‹Γ = Γ'› by auto
          then have Γ ⇒ C ↓ 0 by auto
          then show ∃ n. Γ ⇒ C ↓ n by blast
      next
          case (ConjR Γ1 G k H l)
          then have Γ' ⊕ E⊃F ⇒ G ↓ k and Γ' ⊕ E⊃F ⇒ H ↓ l by auto
          from ‹Γ ⇒ A ↓ n› and ‹A = E⊃F› and ‹Γ = Γ'› have Γ' ⇒ E⊃F ↓ n by simp
          with ‹Γ' ⊕ E⊃F ⇒ G ↓ k› have ∃ n. Γ' ⇒ G ↓ n
              using IH[where Γ=Γ' and A=E⊃F and C=G and n=n and m=k]
              and ‹m = k+l+1› and ‹x = length A› and ‹A = E ⊃ F› and ‹y = n+m+1›
              by auto
          moreover from ‹Γ' ⇒ E⊃F ↓ n› and ‹Γ' ⊕ E⊃F ⇒ H ↓ l› have ∃ n. Γ' ⇒ H ↓ n
              using IH[where Γ=Γ' and A=E⊃F and C=H and n=n and m=l]
              and ‹m = k+l+1› and ‹x = length A› and ‹A = E ⊃ F› and ‹y = n+m+1›
              by auto
          ultimately have ∃ n. Γ' ⇒ G∧*H ↓ n by auto
          with ‹Γ = Γ'› and ‹C = G∧*H› show ∃ n. Γ ⇒ C ↓ n by auto
      next
          case (DisjR1 Γ1 G k H)
          then have Γ' ⊕ E⊃F ⇒ G ↓ k by auto
          moreover from ‹Γ ⇒ A ↓ n› and ‹A = E⊃F› and ‹Γ=Γ'› have Γ' ⇒ E⊃F ↓ n by
```

simp
ultimately have $\exists\ n.\ \Gamma' \Rightarrow G \downarrow n$
using *IH*[**where** $\Gamma{=}\Gamma'$ **and** $A{=}E{\supset}F$ **and** $n{=}n$ **and** $C{=}G$ **and** $m{=}k$] **and** ‹x = *length A*›
and ‹$A = E{\supset}F$› **and** ‹$y = n{+}m{+}1$› **and** ‹$m{=}k{+}1$› **by** *auto*
with ‹$\Gamma = \Gamma'$› **and** ‹$C = G{\vee}{*}H$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*DisjR2* $\Gamma 1\ H\ k\ G$)
then have $\Gamma' \oplus E{\supset}F \Rightarrow H \downarrow k$ **by** *auto*
moreover from ‹$\Gamma \Rightarrow A \downarrow n$› **and** ‹$A = E{\supset}F$› **and** ‹$\Gamma{=}\Gamma'$› **have** $\Gamma' \Rightarrow E{\supset}F \downarrow n$ **by** *simp*
ultimately have $\exists\ n.\ \Gamma' \Rightarrow H \downarrow n$
using *IH*[**where** $\Gamma{=}\Gamma'$ **and** $A{=}E{\supset}F$ **and** $n{=}n$ **and** $C{=}H$ **and** $m{=}k$] **and** ‹x = *length A*›
and ‹$A = E{\supset}F$› **and** ‹$y = n{+}m{+}1$› **and** ‹$m{=}k{+}1$› **by** *auto*
with ‹$\Gamma = \Gamma'$› **and** ‹$C = G{\vee}{*}H$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*ImpR* $\Gamma 1\ G\ H\ k$)
then have $\Gamma' \oplus G \oplus E{\supset}F \Rightarrow H \downarrow k$ **by** (*auto simp add*:*union-ac*)
moreover from ‹$\Gamma \Rightarrow A \downarrow n$› **and** ‹$A = E{\supset}F$› **and** ‹$\Gamma{=}\Gamma'$› **have** $\Gamma' \Rightarrow E{\supset}F \downarrow n$ **by** *simp*
then have $\Gamma' \oplus G \Rightarrow E{\supset}F \downarrow n$ **using** *dpWeak* **by** *auto*
ultimately have $\exists\ n.\ \Gamma' \oplus G \Rightarrow H \downarrow n$
using *IH*[**where** $\Gamma{=}\Gamma'{\oplus}G$ **and** $A{=}E{\supset}F$ **and** $n{=}n$ **and** $C{=}H$ **and** $m{=}k$] **and** ‹x = *length A*›
and ‹$A = E{\supset}F$› **and** ‹$y{=}n{+}m{+}1$› **and** ‹$m{=}k{+}1$› **by** *auto*
with ‹$\Gamma = \Gamma'$› **and** ‹$C = G{\supset}H$› **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*ConjL* $\Gamma 1\ G\ H\ C'\ k$)
then obtain $\Gamma 2$ **where** $\Gamma' = \Gamma 2 \oplus G{\wedge}{*}H$
and $\Gamma 1 = \Gamma 2 \oplus E{\supset}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $A{=}E{\supset}F$ **and** $\Gamma'{=}\Gamma 1$ **and** $B{=}G{\wedge}{*}H$]
by *auto*
with ‹$\Gamma 1 \oplus G \oplus H \Rightarrow C' \downarrow k$› **have** $\Gamma 2 \oplus G \oplus H \oplus E{\supset}F \Rightarrow C' \downarrow k$ **by** (*auto simp add*:*union-ac*)
moreover from ‹$\Gamma \Rightarrow A \downarrow n$› **and** ‹$\Gamma = \Gamma'$› **and** ‹$A = E{\supset}F$› **and** ‹$\Gamma' = \Gamma 2 \oplus G{\wedge}{*}H$›
have $\Gamma 2 \oplus G{\wedge}{*}H \Rightarrow E{\supset}F \downarrow n$ **by** *auto*
then obtain n' **where** $n'{\leq}n$ **and** $\Gamma 2 \oplus G \oplus H \Rightarrow E{\supset}F \downarrow n'$
using *inversionConjL*[**where** $\Gamma{=}\Gamma 2$ **and** $A{=}G$ **and** $B{=}H$] **by** *auto*
ultimately have $\exists\ n.\ \Gamma 2 \oplus G \oplus H \Rightarrow C' \downarrow n$ **using** ‹$y = n{+}m{+}1$› **and** ‹$m{=}k{+}1$› **and** ‹x = *length A*› **and** ‹$A = E{\supset}F$›
and *IH*[**where** $\Gamma{=}\Gamma 2{\oplus}G{\oplus}H$ **and** $A{=}E{\supset}F$ **and** $n{=}n'$ **and** $m{=}k$ **and** $C{=}C'$] **by** *auto*

then have $\exists\ n.\ \Gamma 2 \oplus G \wedge * H \Rightarrow C' \downarrow n$ **using** *provable-dp.ConjL* **by** *auto*
with $\langle \Gamma' = \Gamma 2 \oplus G \wedge * H \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** $\langle C = C' \rangle$ **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*DisjL* $\Gamma 1\ G\ C'\ k\ H\ l$)
then obtain $\Gamma 2$ **where** $\Gamma' = \Gamma 2 \oplus G \vee * H$
and $\Gamma 1 = \Gamma 2 \oplus E \supset F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $A = E \supset F$] **by** *auto*
with $\langle \Gamma 1 \oplus G \Rightarrow C' \downarrow k \rangle$ **and** $\langle \Gamma 1 \oplus H \Rightarrow C' \downarrow l \rangle$ **have**
$\Gamma 2 \oplus G \oplus E \supset F \Rightarrow C' \downarrow k$ **and** $\Gamma 2 \oplus H \oplus E \supset F \Rightarrow C' \downarrow l$ **by** (*auto simp add*:*union-ac*)
from $\langle \Gamma \Rightarrow A \downarrow n \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** $\langle A = E \supset F \rangle$ **and** $\langle \Gamma' = \Gamma 2 \oplus G \vee * H \rangle$
have $\Gamma 2 \oplus G \vee * H \Rightarrow E \supset F \downarrow n$ **by** *auto*
then obtain *n1 n2* **where** $n1 \leq n$ **and** $n2 \leq n$ **and**
$\Gamma 2 \oplus G \Rightarrow E \supset F \downarrow n1$ **and**
$\Gamma 2 \oplus H \Rightarrow E \supset F \downarrow n2$
using *inversionDisjL*[**where** $\Gamma = \Gamma 2$ **and** $A = G$ **and** $B = H$ **and** $n = n$ **and** $C = E \supset F$] **by** *auto*
from $\langle \Gamma 2 \oplus G \Rightarrow E \supset F \downarrow n1 \rangle$ **and** $\langle \Gamma 2 \oplus G \oplus E \supset F \Rightarrow C' \downarrow k \rangle$
have $\exists\ n.\ \Gamma 2 \oplus G \Rightarrow C' \downarrow n$ **using** $\langle n1 \leq n \rangle$ **and** $\langle m = k + l + 1 \rangle$
and *IH*[**where** $\Gamma = \Gamma 2 \oplus G$ **and** $A = E \supset F$ **and** $C = C'$ **and** $n = n1$ **and** $m = k$] **and** $\langle x = \mathit{length}\ A \rangle$
and $\langle A = E \supset F \rangle$ **and** $\langle y = n + m + 1 \rangle$ **by** *auto*
moreover from $\langle \Gamma 2 \oplus H \Rightarrow E \supset F \downarrow n2 \rangle$ **and** $\langle \Gamma 2 \oplus H \oplus E \supset F \Rightarrow C' \downarrow l \rangle$
have $\exists\ n.\ \Gamma 2 \oplus H \Rightarrow C' \downarrow n$ **using** $\langle n2 \leq n \rangle$ **and** $\langle m = k + l + 1 \rangle$
and *IH*[**where** $\Gamma = \Gamma 2 \oplus H$ **and** $A = E \supset F$ **and** $C = C'$ **and** $n = n2$ **and** $m = l$] **and** $\langle x = \mathit{length}\ A \rangle$
and $\langle A = E \supset F \rangle$ **and** $\langle y = n + m + 1 \rangle$ **by** *auto*
ultimately have $\exists\ n.\ \Gamma 2 \oplus G \vee * H \Rightarrow C' \downarrow n$ **by** *auto*
with $\langle \Gamma' = \Gamma 2 \oplus G \vee * H \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** $\langle C = C' \rangle$ **show** $\exists\ n.\ \Gamma \Rightarrow C \downarrow n$ **by** *auto*
next
case (*ImpL* $\Gamma 1\ G\ H\ k\ C'\ l$)
have $E \supset F = G \supset H \vee E \supset F \neq G \supset H$ **by** *blast*
moreover
{**assume** $E \supset F \neq G \supset H$
with $\langle \Gamma' \oplus E \supset F = \Gamma 1 \oplus G \supset H \rangle$ **obtain** $\Gamma 2$ **where**
$\Gamma' = \Gamma 2 \oplus G \supset H$ **and**
$\Gamma 1 = \Gamma 2 \oplus E \supset F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $A = E \supset F$] **by** *auto*
with $\langle \Gamma \Rightarrow A \downarrow n \rangle$ **and** $\langle A = E \supset F \rangle$ **and** $\langle \Gamma = \Gamma' \rangle$ **and** $\langle \Gamma 1 \oplus H \Rightarrow C' \downarrow l \rangle$
have $\Gamma 2 \oplus E \supset F \oplus H \Rightarrow C' \downarrow l$ **and** $\Gamma 2 \oplus G \supset H \Rightarrow E \supset F \downarrow n$ **by** (*auto simp add*:*union-ac*)
then obtain n' **where** $n' \leq n$ **and** $\Gamma 2 \oplus H \Rightarrow E \supset F \downarrow n'$
using *inversionImpL*[**where** $\Gamma = \Gamma 2$ **and** $A = G$ **and** $B = H$ **and** $C = E \supset F$ **and** $n = n$] **by** *auto*

```
            with ‹Γ2 ⊕ E⊃F ⊕ H ⇒ C' ↓ l› have ∃ n. Γ2 ⊕ H ⇒ C' ↓ n
                using IH[where Γ=Γ2⊕H and A=E⊃F and C=C' and n=n' and m=l]
                 and ‹x = length A› and ‹A = E⊃F› and ‹y = n+m+1› and ‹m=k+l+1› by
(auto simp add:union-ac)
            moreover from ‹Γ1 ⊕ G⊃H ⇒ G ↓ k› and ‹Γ1 = Γ2 ⊕ E⊃F›
                have Γ2 ⊕ G⊃H ⊕ E⊃F ⇒ G ↓ k by (auto simp add:union-ac)
            with ‹Γ2 ⊕ G⊃H ⇒ E⊃F ↓ n› have ∃ n. Γ2 ⊕ G⊃H ⇒ G ↓ n
               using IH[where Γ=Γ2 ⊕ G⊃H and A=E⊃F and C=G and n=n and m=k]
                and ‹x = length A› and ‹A = E⊃F› and ‹y = n+m+1› and ‹m=k+l+1› by
auto
            ultimately have ∃ n. Γ2 ⊕ G⊃H ⇒ C' ↓ n by auto
           with ‹Γ = Γ'› and ‹Γ' = Γ2 ⊕ G⊃H› and ‹C=C'› have ∃ n. Γ ⇒ C ↓ n by auto
           }
         moreover
           {assume E⊃F = G⊃H
            then have E = G and F=H by auto
            with ‹Γ' ⊕ E⊃F = Γ1 ⊕ G⊃H› have Γ' = Γ1 by auto
             with ‹Γ = Γ'› and ‹Γ' ⊕ E ⇒ F ↓ j› and ‹E=G› and ‹F=H› and ‹Γ ⇒ A ↓ n›
and ‹A = E⊃F›
                have Γ1 ⊕ G ⇒ H ↓ j and Γ1 ⇒ G⊃H ↓ n by auto
            with ‹Γ1 ⊕ G⊃H ⇒ G ↓ k› have ∃ n. Γ1 ⇒ G ↓ n
                using IH[where Γ=Γ1 and A=G⊃H and C=G and n=n and m=k]
                 and ‹x = length A› and ‹A = E⊃F› and ‹E⊃F = G⊃H› and ‹y = n+m+1›
and ‹m=k+l+1› by auto
            then obtain n' where Γ1 ⇒ G ↓ n' by blast
            with ‹Γ1 ⊕ G ⇒ H ↓ j› have ∃ n. Γ1 ⇒ H ↓ n
                using IH[where Γ=Γ1 and A=G and C=H and n=n' and m=j]
                and ‹x = length A› and ‹A = E⊃F› and ‹E⊃F = G⊃H› by auto
            then obtain m' where Γ1 ⇒ H ↓ m' by blast
            with ‹Γ1 ⊕ H ⇒ C' ↓ l› have ∃ n. Γ1 ⇒ C' ↓ n
                using IH[where Γ=Γ1 and A=H and C=C' and n=m' and m=l]
                and ‹x = length A› and ‹A = E⊃F› and ‹E⊃F = G⊃H› by auto
               with ‹Γ' = Γ1› and ‹Γ = Γ'› and ‹C = C'› have ∃ n. Γ ⇒ C ↓ n by auto

           }
         ultimately show ∃ n. Γ ⇒ C ↓ n by blast
       qed
     qed
qed


lemma contextSplitCut:
  assumes Γ ⇒ A ↓ n and Γ' ⊕ A ⇒ C ↓ m
```

shows $\exists\ a.\ \Gamma + \Gamma' \Rightarrow C \downarrow a$
proof−
from *assms* **have** $\Gamma + \Gamma' \Rightarrow A \downarrow n$ **using** *dpWeak'* **by** *auto*
moreover
from *assms* **have** $\Gamma' \oplus A + \Gamma \Rightarrow C \downarrow m$ **using** *dpWeak'*[**where** $\Gamma{=}\Gamma'{\oplus}A$ **and** $\Gamma'{=}\Gamma$] **by** *auto*
then have $\Gamma + \Gamma' \oplus A \Rightarrow C \downarrow m$ **by** (*simp only*:*union-ac*)
ultimately
show $\exists\ a.\ \Gamma + \Gamma' \Rightarrow C \downarrow a$ **using** *cutAdmissibility*[**where** $\Gamma{=}\Gamma{+}\Gamma'$] **by** (*auto simp add*:*union-ac*)
qed

lemma *genAx*:
shows $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$
proof (*induct A*)
case (*Atom i*)
then have $\Gamma \oplus Atom\ i \Rightarrow Atom\ i \downarrow 0$ **using** *Ax* **by** *auto*
then show *?case* **by** *blast*
next
case *ff*
then have $\Gamma \oplus ff \Rightarrow ff \downarrow 0$ **using** *LBot* **by** *auto*
then show *?case* **by** *blast*
next
case (*Conj A B*)
then obtain *n m* **where** *l*: $\Gamma \oplus A \Rightarrow A \downarrow n$
and *r*: $\Gamma \oplus B \Rightarrow B \downarrow m$ **by** *auto*
from *l* **have** $\Gamma \oplus A \oplus B \Rightarrow A \downarrow n$ **using** *dpWeak*[**where** $\Gamma{=}\Gamma{\oplus}A$ **and** $A{=}B$] **by** *auto*
then have $\Gamma \oplus A{\wedge}{*}B \Rightarrow A \downarrow n{+}1$ **by** (*rule ConjL*)
moreover
from *r* **have** $\Gamma \oplus A \oplus B \Rightarrow B \downarrow m$ **using** *dpWeak*[**where** $\Gamma{=}\Gamma{\oplus}B$ **and** $A{=}A$] **by** (*auto simp add*:*union-ac*)
then have $\Gamma \oplus A{\wedge}{*}B \Rightarrow B \downarrow m{+}1$ **by** (*rule ConjL*)
ultimately
have $\Gamma \oplus A{\wedge}{*}B \Rightarrow A{\wedge}{*}B \downarrow n{+}1{+}(m{+}1)\ {+}1$
using *provable-dp.ConjR*[**where** $\Gamma{=}\Gamma{\oplus}A{\wedge}{*}B$ **and** $n{=}n{+}1$ **and** $m{=}m{+}1$ **and** $A{=}A$ **and** $B{=}B$] **by** *simp*
then show *?case* **by** *blast*
next
case (*Disj A B*)
then obtain *n m* **where** *l*: $\Gamma \oplus A \Rightarrow A \downarrow n$
and *r*: $\Gamma \oplus B \Rightarrow B \downarrow m$ **by** *auto*
from *l* **have** $\Gamma \oplus A \Rightarrow A{\vee}{*}B \downarrow n{+}1$ **by** (*rule DisjR1*)
moreover
from *r* **have** $\Gamma \oplus B \Rightarrow A{\vee}{*}B \downarrow m{+}1$ **by** (*rule DisjR2*)
ultimately

have $\Gamma \oplus A \vee * B \Rightarrow A \vee * B \downarrow n{+}1{+}(m{+}1)\ {+}1$ **using** *provable-dp.DisjL*[**where** $C{=}A \vee * B$ **and** $n{=}n{+}1$ **and** $m{=}m{+}1$] **by** *simp*
then show *?case* **by** *blast*
next
case (*Imp A B*)
then obtain $n\ m$ **where** l: $\Gamma \oplus A \Rightarrow A \downarrow n$
and r: $\Gamma \oplus B \Rightarrow B \downarrow m$ **by** *auto*
from l **have** $\Gamma \oplus A \oplus A \supset B \Rightarrow A \downarrow n$ **using** *dpWeak*[**where** $\Gamma{=}\Gamma \oplus A$ **and** $A{=}A \supset B$] **by** *auto*
moreover
from r **have** $\Gamma \oplus B \oplus A \Rightarrow B \downarrow m$ **using** *dpWeak*[**where** $\Gamma{=}\Gamma \oplus B$ **and** $A{=}A$] **by** (*auto simp add*:*union-ac*)
ultimately
have $\Gamma \oplus A \supset B \oplus A \Rightarrow B \downarrow n{+}m{+}1$ **using** *provable-dp.ImpL*[**where** $\Gamma{=}\Gamma \oplus A$ **and** $C{=}B$ **and** $A{=}A$ **and** $B{=}B$] **by** (*auto simp add*:*union-ac*)
then have $\Gamma \oplus A \supset B \Rightarrow A \supset B \downarrow n{+}m{+}2$ **using** *provable-dp.ImpR*[**where** $\Gamma{=}\Gamma \oplus A \supset B$ **and** $A{=}A$ **and** $B{=}B$] **by** *auto*
then show *?case* **by** *blast*
qed

end

D.2 Contraction Admissibility for G4ip

This file uses Multiset.thy, which is included in the *Isabelle* distribution.

datatype *form* = *At nat*
| *Imp form form* (-⊃- [*100*,*100*]*110*)
| *Conj form form* (- ∧* - [*100*,*100*]*110*)
| *Disj form form* (- ∨* - [*100*,*100*]*110*)
| *ff*

abbreviation
multiset-plus (**infixl** ⊕ *80*) **where**
(Γ :: *form multiset*) ⊕ (*A* :: *form*) ≡ Γ + {#*A*#}
abbreviation
multiset-minus (**infixl** ⊖ *80*) **where**
(Γ :: *form multiset*) ⊖ (*A* :: *form*) ≡ Γ − {#*A*#}

inductive
provable :: *form multiset* ⇒ *form* ⇒ *nat* ⇒ *bool* (- ⇒ - ↓ - [*60*,*60*,*60*] *60*)
where
Ax[*intro*]: ⟦ (*At i*):# Γ⟧ ⟹ Γ ⇒ *At i* ↓ *0*

| *LBot*[*intro*]: $[\![\mathit{ff} :\# \Gamma]\!] \Longrightarrow \Gamma \Rightarrow C \downarrow 0$
| *ConjR*[*intro*]: $[\![\Gamma \Rightarrow A \downarrow n \,;\, \Gamma \Rightarrow B \downarrow m]\!] \Longrightarrow \Gamma \Rightarrow A \wedge* B \downarrow n{+}m{+}1$
| *ConjL*[*intro*]: $[\![\Gamma \oplus A \oplus B \Rightarrow C \downarrow n]\!] \Longrightarrow \Gamma \oplus A{\wedge*}B \Rightarrow C \downarrow n{+}1$
| *DisjR1*[*intro*]: $[\![\Gamma \Rightarrow A \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A{\vee*}B \downarrow n{+}1$
| *DisjR2*[*intro*]: $[\![\Gamma \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A{\vee*}B \downarrow n{+}1$
| *DisjL*[*intro*]: $[\![\Gamma \oplus A \Rightarrow C \downarrow n \,;\, \Gamma \oplus B \Rightarrow C \downarrow m]\!] \Longrightarrow \Gamma \oplus A{\vee*}B \Rightarrow C \downarrow n{+}m{+}1$
| *ImpR*[*intro*]: $[\![\, \Gamma \oplus A \Rightarrow B \downarrow n]\!] \Longrightarrow \Gamma \Rightarrow A{\supset}B \downarrow n{+}1$
| *ImpL0*[*intro*]: $[\![\Gamma \oplus \mathit{At}\ i \oplus B \Rightarrow C \downarrow n]\!] \Longrightarrow \Gamma \oplus \mathit{At}\ i \oplus (\mathit{At}\ i \supset B) \Rightarrow C \downarrow n{+}1$
| *ImpLC*[*intro*]: $[\![\Gamma \oplus A \supset (B \supset C) \Rightarrow D \downarrow n]\!] \Longrightarrow \Gamma \oplus (A{\wedge*}B) \supset C \Rightarrow D \downarrow n{+}1$
| *ImpLD*[*intro*]: $[\![\Gamma \oplus A \supset C \oplus B \supset C \Rightarrow D \downarrow n]\!] \Longrightarrow \Gamma \oplus (A{\vee*}B) \supset C \Rightarrow D \downarrow n{+}1$
| *ImpLL*[*intro*]: $[\![\Gamma \oplus A \oplus B \supset C \Rightarrow B \downarrow n \,;\, \Gamma \oplus C \Rightarrow D \downarrow m]\!] \Longrightarrow \Gamma \oplus (A{\supset}B){\supset}C \Rightarrow D \downarrow n{+}m{+}1$

consts *weight* :: *form* ⇒ *nat*
primrec
  weight (*At i*) = *1*
  weight (*A* ⊃ *B*) = *1* + *weight A* + *weight B*
  weight (*A* ∧* *B*) = *2* + *weight A* + *weight B*
  weight (*A* ∨* *B*) = *3* + *weight A* + *weight B*
  weight (*ff*) = *0*

abbreviation
less-prod-nat (- <* - [*50*,*50*]*50*) **where**
  p <* *q* ≡ (*p*,*q*) : *less-than* <**lex**> *less-than*

lemma *nat-prod-induct* [*case-names less*]:
 fixes *x y* :: *nat*
 assumes *induct-step*: ⋀ *x y*. (⋀ *u v*. (*u*, *v*) <* (*x*, *y*) ⟹ *P u v*) ⟹ *P x y*
 shows *P x y*
proof −
 have *wf* (*less-than* <**lex**> *less-than*) **by** *blast*
 then show *?thesis*
 proof (*induct p* ≡ (*x*, *y*) *arbitrary*: *x y*)
  case (*less p*)
  show *P x y*
  proof (*rule induct-step*)
   fix *u v*
   assume (*u*, *v*) <* (*x*, *y*)
   with *less* **show** *P u v* **by** *simp*
  qed
 qed
qed

lemma *containMultiset*:
assumes $A :\# \Gamma$
shows $\exists\ \Gamma'.\ \Gamma = \Gamma' \oplus A$
proof −
from *assms* **have** $\Gamma \ominus A \oplus A = \Gamma$ **by** (*auto simp add*:*multiset-eq-conv-count-eq*)
then show $\exists\ \Gamma'.\ \Gamma = \Gamma' \oplus A$ **by** (*rule-tac x*=$\Gamma \ominus A$ **in** *exI*) (*auto*)
qed

lemma *midMultiset*:
assumes $\Gamma \oplus A = \Gamma' \oplus B$ **and** $A \neq B$
shows $\exists\ \Gamma''.\ \Gamma = \Gamma'' \oplus B \wedge \Gamma' = \Gamma'' \oplus A$
proof −
from *assms* **have** $A :\# \Gamma'$
proof −
from *assms* **have** *set-of* $(\Gamma \oplus A) =$ *set-of* $(\Gamma' \oplus B)$ **by** *auto*
then have *set-of* $\Gamma \cup \{A\} =$ *set-of* $\Gamma' \cup \{B\}$ **by** *auto*
then have *set-of* $\Gamma \cup \{A\} \subseteq$ *set-of* $\Gamma' \cup \{B\}$ **by** *simp*
then have $A \in$ *set-of* Γ' **using** *assms* **by** *auto*
thus $A :\# \Gamma'$ **by** *simp*
qed
then have $\exists\ \Gamma''.\ \Gamma' = \Gamma'' \oplus A$ **using** *containMultiset*[**where** Γ=Γ'] **by** *auto*
then obtain Γ'' **where** *eq1*:$\Gamma' = \Gamma'' \oplus A$ **by** *blast*
from ‹$\Gamma \oplus A = \Gamma' \oplus B$› *eq1* **have** $\Gamma \oplus A = \Gamma'' \oplus A \oplus B$ **by** *auto*
then have $\Gamma = \Gamma'' \oplus B$ **by** (*auto simp add*:*multiset-eq-conv-count-eq*)
thus *?thesis* **using** *eq1* **by** *blast*
qed

lemma *midMultiset2*:
assumes $\Gamma \oplus A \oplus A = \Gamma' \oplus B$ **and** $A \neq B$
shows $\exists\ \Gamma''.\ \Gamma = \Gamma'' \oplus B \wedge \Gamma' = \Gamma'' \oplus A \oplus A$
proof −
from *assms* **have** $\exists\ \Gamma 1.\ \Gamma \oplus A = \Gamma 1 \oplus B \wedge \Gamma' = \Gamma 1 \oplus A$ **using** *midMultiset*[**where** Γ=$\Gamma \oplus A$]
by *auto*
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma \oplus A = \Gamma 1 \oplus B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus A$ **by** *blast*
from *eq1* **have** $\exists\ \Gamma 2.\ \Gamma = \Gamma 2 \oplus B \wedge \Gamma 1 = \Gamma 2 \oplus A$ **using** ‹$A \neq B$› **and** *midMultiset* **by** *auto*
then obtain $\Gamma 2$ **where** *eq3*: $\Gamma = \Gamma 2 \oplus B$ **and** *eq4*: $\Gamma 1 = \Gamma 2 \oplus A$ **by** *blast*
from *eq2* **and** *eq4* **have** $\Gamma' = \Gamma 2 \oplus A \oplus A$ **by** *simp*
then show $\exists\ \Gamma''.\ \Gamma = \Gamma'' \oplus B \wedge \Gamma' = \Gamma'' \oplus A \oplus A$ **using** *eq3* **by** (*rule-tac x*=$\Gamma 2$ **in** *exI*) (*auto*)
qed

lemma *dpWeak*:
fixes Γ :: *form multiset*
assumes *a*: $\Gamma \Rightarrow C \downarrow n$
shows $\Gamma \oplus A \Rightarrow C \downarrow n$
using *a*
proof (*induct* Γ *C n*)
case (*Ax i* Γ)
then have *At i* :# Γ **by** *simp-all*
then show $\Gamma \oplus A \Rightarrow At\ i \downarrow 0$ **using** *provable.Ax* **by** *auto*
next
case (*LBot* Γ *C*)
then have *ff* :# Γ **by** *simp-all*
then show $\Gamma \oplus A \Rightarrow C \downarrow 0$ **using** *provable.LBot* **by** *auto*
next
case (*ConjR* Γ *C n D m*)
then have $\Gamma \oplus A \Rightarrow C \wedge * D \downarrow n+m+1$ **using** *provable.ConjR*[**where** Γ=$\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR1* Γ *C n D*)
then have $\Gamma \oplus A \Rightarrow C \vee * D \downarrow n+1$ **using** *provable.DisjR1*[**where** Γ=$\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR2* Γ *D n C*)
then have $\Gamma \oplus A \Rightarrow C \vee * D \downarrow n+1$ **using** *provable.DisjR2*[**where** Γ=$\Gamma \oplus A$] **by** *auto*
then show *?case* **by** *blast*
next
case (*ImpR* Γ *C D m*)
then have $\Gamma \oplus C \oplus A \Rightarrow D \downarrow m$ **by** *auto*
then show $\Gamma \oplus A \Rightarrow C \supset D \downarrow m+1$ **using** *provable.ImpR* **by** (*auto simp add*:*union-ac*)
next
case (*ConjL* Γ *C D E n*)
then have $\Gamma \oplus C \wedge * D \oplus A \Rightarrow E \downarrow n+1$ **using** *provable.ConjL*[**where** Γ=$\Gamma \oplus A$ **and** A=C **and** B=D] **by** (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*DisjL* Γ *C E n D m*)
then have $\Gamma \oplus C \vee * D \oplus A \Rightarrow E \downarrow n+m+1$ **using** *provable.DisjL*[**where** Γ=$\Gamma \oplus A$ **and** A=C **and** B=D] **by** (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*ImpL0* Γ *i B C n*)
then have $\Gamma \oplus At\ i \oplus B \oplus A \Rightarrow C \downarrow n$ **by** *simp*

then show $\Gamma \oplus At\ i \oplus (At\ i \supset B) \oplus A \Rightarrow C \downarrow n{+}1$
using *provable.ImpL0*[**where** $C{=}C$ **and** $B{=}B$ **and** $\Gamma{=}\Gamma{\oplus}A$ **and** $i{=}i$ **and** $n{=}n$] **by** (*auto simp add*:*union-ac*)
next
case (*ImpLC* Γ C D B E n)
then have $\Gamma \oplus C{\supset}(D{\supset}B) \oplus A \Rightarrow E \downarrow n$ **by** *simp*
then show $\Gamma \oplus (C{\wedge}{*}D){\supset}B \oplus A \Rightarrow E \downarrow n{+}1$
using *provable.ImpLC*[**where** $\Gamma{=}\Gamma{\oplus}A$ **and** $A{=}C$ **and** $B{=}D$ **and** $C{=}B$ **and** $D{=}E$ **and** $n{=}n$] **by** (*auto simp add*:*union-ac*)
next
case (*ImpLD* Γ C B D E n)
then have $\Gamma \oplus C{\supset}B \oplus D{\supset}B \oplus A \Rightarrow E \downarrow n$ **by** *simp*
then show $\Gamma \oplus (C{\vee}{*}D) \supset B \oplus A \Rightarrow E \downarrow n{+}1$
using *provable.ImpLD*[**where** $\Gamma{=}\Gamma{\oplus}A$ **and** $A{=}C$ **and** $B{=}D$ **and** $C{=}B$ **and** $D{=}E$ **and** $n{=}n$] **by** (*auto simp add*:*union-ac*)
next
case (*ImpLL* Γ C D B n E m)
then have $\Gamma \oplus C \oplus D{\supset}B \oplus A \Rightarrow D \downarrow n$ **and** $\Gamma \oplus B \oplus A \Rightarrow E \downarrow m$ **by** *auto*
then show $\Gamma \oplus (C{\supset}D){\supset}B \oplus A \Rightarrow E \downarrow n{+}m{+}1$
using *provable.ImpLL*[**where** $\Gamma{=}\Gamma{\oplus}A$ **and** $A{=}C$ **and** $B{=}D$ **and** $C{=}B$ **and** $D{=}E$ **and** $n{=}n$ **and** $m{=}m$] **by** (*auto simp add*:*union-ac*)
qed

lemma *dpWeak'*:
assumes $\Gamma \Rightarrow C \downarrow n$
shows $\Gamma + \Gamma' \Rightarrow C \downarrow n$
using *assms*
proof (*induct* Γ')
case *empty*
then show *?case* **by** *auto*
next
case (*add* Γ' x)
then have $\Gamma + \Gamma' \Rightarrow C \downarrow n$ **by** *auto*
then have $\Gamma + \Gamma' \oplus x \Rightarrow C \downarrow n$ **using** *dpWeak*[**where** $\Gamma{=}\Gamma{+}\Gamma'$ **and** $A{=}x$] **by** (*auto simp only*:*union-ac*)
then show *?case* **by** *blast*
qed

lemma *inversionConjL*:
assumes $\Gamma \oplus A{\wedge}{*}B \Rightarrow C \downarrow n$
shows $\exists\ j.\ j{\leq}n \wedge \Gamma \oplus A \oplus B \Rightarrow C \downarrow j$
using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma \oplus A{\wedge}{*}B$ C n *arbitrary*: Γ)

case (*Ax i* Γ′)
then have *At i* :# Γ **by** *auto*
then have Γ ⊕ *A* ⊕ *B* ⇒ *At i* ↓ *0* **by** *auto*
then show *?case* **by** *blast*
next
case (*LBot* Γ′ *C*)
then have *ff* :# Γ **by** *auto*
then have Γ ⊕ *A* ⊕ *B* ⇒ *C* ↓ *0* **by** *auto*
then show *?case* **by** *blast*
next
case (*ImpR* Γ′ *E F k*)
then have Γ′ ⊕ *E* = Γ ⊕ *A*∧∗*B* ⊕ *E* **by** *auto*
then have ∃ *j*. *j*≤*k* ∧ Γ ⊕ *A* ⊕ *B* ⊕ *E* ⇒ *F* ↓ *j* **using** *prems*(*3*)[**where** Γ=Γ⊕*E*] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *c1*: *j*≤*k*
and *c2*: Γ ⊕ *A* ⊕ *B* ⊕ *E* ⇒ *F* ↓ *j* **by** *auto*
from *c2* **have** Γ ⊕ *A* ⊕ *B* ⇒ *E*⊃*F* ↓ *j*+*1* **using** *provable.ImpR*[**where** Γ=Γ ⊕ *A* ⊕ *B* **and** *A*=*E* **and** *B*=*F*] **by** *auto*
then show *?case* **using** *c1* **by** *auto*
next
case (*ConjR* Γ′ *E k F l*)
then have ∃*j*≤*k*. Γ ⊕ *A* ⊕ *B* ⇒ *E* ↓ *j* **and** ∃*j*≤*l*. Γ ⊕ *A* ⊕ *B* ⇒ *F* ↓ *j* **by** *auto*
then obtain *j1 j2* **where** *c1*: *j1* ≤ *k*
and *c2*: Γ ⊕ *A* ⊕ *B* ⇒ *E* ↓ *j1*
and *c3*: *j2* ≤ *l*
and *c4*: Γ ⊕ *A* ⊕ *B* ⇒ *F* ↓ *j2* **by** *auto*
then show *?case* **using** *provable.ConjR*[**where** Γ=Γ⊕*A*⊕*B* **and** *n*=*j1* **and** *m*=*j2* **and** *A*=*E* **and** *B*=*F*]
apply (*rule-tac x*=*j1*+*j2*+*1* **in** *exI*) **by** *auto*
next
case (*DisjR1* Γ′ *E n F*)
then have ∃*j*≤*n*. Γ ⊕ *A* ⊕ *B* ⇒ *E* ↓ *j* **by** *auto*
then obtain *j* **where** *eq*:*j*≤*n* **and** Γ ⊕ *A* ⊕ *B* ⇒ *E* ↓ *j* **by** *blast*
then have Γ ⊕ *A* ⊕ *B* ⇒ *E*∨∗*F* ↓ *j*+*1* **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ′ *F n E*)
then have ∃*j*≤*n*. Γ ⊕ *A* ⊕ *B* ⇒ *F* ↓ *j* **by** *auto*
then obtain *j* **where** *eq*:*j*≤*n* **and** Γ ⊕ *A* ⊕ *B* ⇒ *F* ↓ *j* **by** *blast*
then have Γ ⊕ *A* ⊕ *B* ⇒ *E*∨∗*F* ↓ *j*+*1* **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjL* Γ′ *E C n F m* Γ′′)

then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \wedge * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \vee * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = E \vee * F$ **and** $B = A \wedge * B$] **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus A \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
moreover
from *eq1 prems*(5)[**where** $\Gamma = \Gamma 1 \oplus F$] **have** $\exists\ k \leq m.\ \Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
ultimately
obtain $j\ k$ **where** *a*: $j \leq n \wedge k \leq m$
and *b*: $\Gamma 1 \oplus E \oplus A \oplus B \Rightarrow C \downarrow j$
and *c*: $\Gamma 1 \oplus F \oplus A \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b c* **have** $\Gamma 1 \oplus E \vee * F \oplus A \oplus B \Rightarrow C \downarrow j+k+1$ **using** *provable.DisjL*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a eq2* **apply** (*rule-tac* $x = j+k+1$ **in** *exI*) **by** *auto*
next
case (*ImpL0* $\Gamma'\ i\ E\ C\ n\ \Gamma''$)
from *prems* **obtain** $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ i = \Gamma 1 \oplus A \wedge * B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ i \supset E$ **using** *midMultiset*[**where** $\Gamma = \Gamma' \oplus At\ i$ **and** $\Gamma' = \Gamma''$ **and** $A = At\ i \supset E$ **and** $B = A \wedge * B$]
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus A \wedge * B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ i$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma 1$ **and** $A = At\ i$ **and** $B = A \wedge * B$]
by *auto*
from *eq2'' eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ i \oplus At\ i \supset E$ **by** *simp*
from *eq1* **have** $\exists\ j.\ j \leq n \wedge \Gamma 2 \oplus At\ i \oplus E \oplus A \oplus B \Rightarrow C \downarrow j$ **using** *prems*(3)[**where** $\Gamma = \Gamma 2 \oplus At\ i \oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j \leq n$
and *b2*: $\Gamma 2 \oplus A \oplus B \oplus At\ i \oplus E \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
from *b2* **have** $\Gamma 2 \oplus A \oplus B \oplus At\ i \oplus At\ i \supset E \Rightarrow C \downarrow j+1$
using *provable.ImpL0*[**where** $\Gamma = \Gamma 2 \oplus A \oplus B$ **and** $i = i$ **and** $B = E$ **and** $C = C$ **and** $n = j$] **by** (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \oplus B \Rightarrow C \downarrow j+1$ **using** *eq2* **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *b1* **by** (*rule-tac* $x = j+1$ **in** *exI*) (*auto*)
next
case (*ImpLC* $\Gamma'\ D\ E\ C\ G\ n\ \Gamma''$)
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \wedge * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D \wedge * E) \supset C$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (D \wedge * E) \supset C$ **and** $B = A \wedge * B$]
by *auto*
from *eq1* **have** $\exists\ j \leq n.\ \Gamma 1 \oplus D \supset (E \supset C) \oplus A \oplus B \Rightarrow G \downarrow j$ **using** *prems*(3)[**where** $\Gamma = \Gamma 1 \oplus D \supset (E \supset C)$]

by (*auto simp only*:*union-ac*)
then obtain *j* **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus D \supset (E \supset C) \oplus A \oplus B \Rightarrow G \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (D \wedge * E) \supset C \oplus A \oplus B \Rightarrow G \downarrow j+1$ **using** *provable.ImpLC*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = D$ **and** $B = E$ **and** $C = C$ **and** $D = G$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \oplus B \Rightarrow G \downarrow j+1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x = j+1$ **in** *exI*) (*auto*)
next
case (*ImpLD* Γ' *D C E G n* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \wedge * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D \vee * E) \supset C$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (D \vee * E) \supset C$ **and** $B = A \wedge * B$]
by *auto*
from *eq1* **have** $\exists\ j \leq n.\ \Gamma 1 \oplus D \supset C \oplus E \supset C \oplus A \oplus B \Rightarrow G \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus D \supset C \oplus E \supset C$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus D \supset C \oplus E \supset C \oplus A \oplus B \Rightarrow G \downarrow j$ **by** *auto*
from *b2* **have** $\Gamma 1 \oplus (D \vee * E) \supset C \oplus A \oplus B \Rightarrow G \downarrow j+1$ **using** *provable.ImpLD*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = D$ **and** $C = C$ **and** $B = E$ **and** $D = G$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \oplus B \Rightarrow G \downarrow j+1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x = j+1$ **in** *exI*) (*auto*)
next
case (*ImpLL* Γ' *D E C n G m* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \wedge * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D \supset E) \supset C$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (D \supset E) \supset C$ **and** $B = A \wedge * B$] **by** *auto*
from *eq1* **have** $\exists\ j \leq n.\ \Gamma 1 \oplus D \oplus E \supset C \oplus A \oplus B \Rightarrow E \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus D \oplus E \supset C$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus D \oplus E \supset C \oplus A \oplus B \Rightarrow E \downarrow j$ **by** *blast*
moreover
from *eq1* **have** $\exists\ k \leq m.\ \Gamma 1 \oplus C \oplus A \oplus B \Rightarrow G \downarrow k$ **using** *prems*(*5*)[**where** $\Gamma = \Gamma 1 \oplus C$] **by** (*auto simp add*:*union-ac*)
then obtain *k* **where** *c1*: $k \leq m$
and *c2*: $\Gamma 1 \oplus C \oplus A \oplus B \Rightarrow G \downarrow k$ **by** *blast*
ultimately
have $\Gamma 1 \oplus (D \supset E) \supset C \oplus A \oplus B \Rightarrow G \downarrow j+k+1$ **using** *provable.ImpLL*[**where** $\Gamma = \Gamma 1 \oplus A \oplus B$ **and** $A = D$ **and** $B = E$ **and** $C = C$ **and** $D = G$ **and** $n = j$ **and** $m = k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \oplus B \Rightarrow G \downarrow j+k+1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1 c1* **by** (*rule-tac* $x = j+k+1$ **in** *exI*) (*auto*)

next
case (*ConjL* Γ' E F C n Γ'')
have $E\wedge *F = A\wedge *B \vee E\wedge *F \neq A\wedge *B$ **by** *blast*
moreover
{assume $E\wedge *F = A\wedge *B$
then have Γ'=Γ'' $\wedge$ E=A $\wedge$ F=B **using** *prems* **by** *auto*
then have $\exists\ j.\ j{\leq}n{+}1\ \wedge\ \Gamma''\oplus A\oplus B \Rightarrow C\downarrow j$ **using** *prems* **apply** (*rule-tac x*=n **in** *exI*) **by** *auto*
}
moreover
{assume $E\wedge *F \neq A\wedge *B$
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A\wedge *B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E\wedge *F$ **using** *midMultiset*[**where** Γ=Γ' **and** Γ'=Γ'' **and** A=$E\wedge *F$ **and** B=$A\wedge *B$] *prems*
by *auto*
from *prems* **have** $\exists\ j.\ j{\leq}n\ \wedge\ \Gamma 1\oplus A\oplus B\oplus E\oplus F \Rightarrow C\downarrow j$ **using** *prems*(*3*)[**where** Γ=$\Gamma 1{\oplus}E{\oplus}F$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1\oplus A\oplus B\oplus E\oplus F \Rightarrow C\downarrow j$ **by** (*auto simp add*:*union-ac*)
from *b2* **have** $\Gamma 1\oplus A\oplus B\oplus E\wedge *F \Rightarrow C\downarrow j{+}1$ **using** *provable.ConjL*[**where** Γ=$\Gamma 1{\oplus}A{\oplus}B$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ j{\leq}n{+}1.\ \Gamma''\oplus A\oplus B \Rightarrow C\downarrow j$ **using** *eq2 b1* **apply** (*rule-tac x*=$j{+}1$ **in** *exI*) **by** (*auto simp add*:*union-ac*)
}
ultimately
show *?case* **by** *blast*
qed

lemma *inversionDisjL*:
assumes $\Gamma\oplus A\vee *B \Rightarrow C\downarrow n$
shows $\exists\ j\ k.\ j{\leq}n\wedge k{\leq}n\wedge \Gamma\oplus A\Rightarrow C\downarrow j\wedge \Gamma\oplus B\Rightarrow C\downarrow k$
using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma{\oplus}A\vee *B$ C n *arbitrary*: Γ)
case (*Ax* i Γ')
then have *At* i :# Γ **by** *auto*
then have $\Gamma\oplus A\Rightarrow$ *At* $i\downarrow 0$ **and** $\Gamma\oplus B\Rightarrow$ *At* $i\downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*LBot* Γ' C)
then have *ff* :# Γ **by** *auto*
then have $\Gamma\oplus A\Rightarrow C\downarrow 0$ **and** $\Gamma\oplus B\Rightarrow C\downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next

case (*ConjR* Γ' *E k F l*)
then have $\exists$ *j1 j2*. $j1{\leq}k \wedge j2{\leq}k \wedge \Gamma \oplus A \Rightarrow E \downarrow j1 \wedge \Gamma \oplus B \Rightarrow E \downarrow j2$
and $\exists$ *j3 j4*. $j3{\leq}l \wedge j4{\leq}l \wedge \Gamma \oplus A \Rightarrow F \downarrow j3 \wedge \Gamma \oplus B \Rightarrow F \downarrow j4$ **by** *auto*
then obtain *j1 j2 j3 j4* **where** *c*: $j1 \leq k \wedge j2{\leq}k \wedge j3{\leq}l \wedge j4{\leq}l$
and *c1*: $\Gamma \oplus A \Rightarrow E \downarrow j1$
and *c2*: $\Gamma \oplus B \Rightarrow E \downarrow j2$
and *c3*: $\Gamma \oplus A \Rightarrow F \downarrow j3$
and *c4*: $\Gamma \oplus B \Rightarrow F \downarrow j4$ **by** *auto*
from *c1 c3* **have** $\Gamma \oplus A \Rightarrow E{\wedge}{*}F \downarrow j1{+}j3{+}1$ **using** *provable.ConjR*[**where** $\Gamma{=}\Gamma{\oplus}A$] **by** *auto*
moreover
from *c2 c4* **have** $\Gamma \oplus B \Rightarrow E{\wedge}{*}F \downarrow j2{+}j4{+}1$ **using** *provable.ConjR*[**where** $\Gamma{=}\Gamma{\oplus}B$] **by** *auto*
ultimately
show *?case* **using** *c* **apply** (*rule-tac x=j1+j3+1* **in** *exI*, *rule-tac x=j2+j4+1* **in** *exI*) **by** *auto*
next
case (*DisjR1* Γ' *E n F*)
then have $\exists$ *j k*. $j{\leq}n \wedge k{\leq}n \wedge \Gamma \oplus A \Rightarrow E \downarrow j \wedge \Gamma \oplus B \Rightarrow E \downarrow k$ **by** *auto*
then obtain *j k* **where** *eq*:$j{\leq}n \wedge k{\leq}n$ **and** $\Gamma \oplus A \Rightarrow E \downarrow j \wedge \Gamma \oplus B \Rightarrow E \downarrow k$ **by** *blast*
then have $\Gamma \oplus A \Rightarrow E{\vee}{*}F \downarrow j{+}1 \wedge \Gamma \oplus B \Rightarrow E{\vee}{*}F \downarrow k{+}1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ' *F n E*)
then have $\exists$ *j k*. $j{\leq}n \wedge k{\leq}n \wedge \Gamma \oplus A \Rightarrow F \downarrow j \wedge \Gamma \oplus B \Rightarrow F \downarrow k$ **by** *auto*
then obtain *j k* **where** *eq*:$j{\leq}n \wedge k{\leq}n$ **and** $\Gamma \oplus A \Rightarrow F \downarrow j \wedge \Gamma \oplus B \Rightarrow F \downarrow k$ **by** *blast*
then have $\Gamma \oplus A \Rightarrow E{\vee}{*}F \downarrow j{+}1 \wedge \Gamma \oplus B \Rightarrow E{\vee}{*}F \downarrow k{+}1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*ImpR* Γ' *E F k*)
then have $\Gamma' \oplus E = \Gamma \oplus A{\vee}{*}B \oplus E$ **by** *auto*
then have $\exists$ *j1 j2*. $j1{\leq}k \wedge j2{\leq}k \wedge \Gamma \oplus A \oplus E \Rightarrow F \downarrow j1 \wedge \Gamma \oplus B \oplus E \Rightarrow F \downarrow j2$
using *prems*(*3*)[**where** $\Gamma{=}\Gamma{\oplus}E$] **by** (*auto simp add*:*union-ac*)
then obtain *j1 j2* **where** *c1*: $j1{\leq}k \wedge j2{\leq}k$
and *c2*: $\Gamma \oplus A \oplus E \Rightarrow F \downarrow j1$
and *c3*: $\Gamma \oplus B \oplus E \Rightarrow F \downarrow j2$ **by** *auto*
from *c2* **have** $\Gamma \oplus A \Rightarrow E{\supset}F \downarrow j1{+}1$ **using** *provable.ImpR*[**where** $\Gamma{=}\Gamma \oplus A$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*
moreover
from *c3* **have** $\Gamma \oplus B \Rightarrow E{\supset}F \downarrow j2{+}1$ **using** *provable.ImpR*[**where** $\Gamma{=}\Gamma \oplus B$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*
ultimately
show *?case* **using** *c1* **apply** (*rule-tac x=j1+1* **in** *exI*, *rule-tac x=j2+1* **in** *exI*) **by** *auto*
next
case (*ConjL* Γ' *E F C n* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A{\vee}{*}B$

and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \wedge * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $A = E \wedge * F$ **and** $\Gamma' = \Gamma''$ **and** $B = A \vee * B$] **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E \oplus F$]
have $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma 1 \oplus E \oplus F \oplus A \Rightarrow C \downarrow j \wedge \Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
then obtain $j\ k$ **where** *eq3*: $j \leq n \wedge k \leq n$
and *c1*: $\Gamma 1 \oplus E \oplus F \oplus A \Rightarrow C \downarrow j$
and *c2*: $\Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *c1* **have** $\Gamma 1 \oplus E \wedge * F \oplus A \Rightarrow C \downarrow j+1$ **using** *provable.ConjL*[**where** $\Gamma = \Gamma 1 \oplus A$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
moreover
from *c2* **have** $\Gamma 1 \oplus E \wedge * F \oplus B \Rightarrow C \downarrow k+1$ **using** *provable.ConjL*[**where** $\Gamma = \Gamma 1 \oplus B$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
ultimately
show *?case* **using** *eq2 eq3* **apply** (*rule-tac* $x = j+1$ **in** *exI*, *rule-tac* $x = k+1$ **in** *exI*) **by** *auto*
next
case (*ImpL0* $\Gamma'\ i\ E\ C\ n\ \Gamma''$)
from *prems* **obtain** $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ i = \Gamma 1 \oplus A \vee * B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ i \supset E$ **using** *midMultiset*[**where** $\Gamma = \Gamma' \oplus At\ i$ **and** $\Gamma' = \Gamma''$ **and** $A = At\ i \supset E$ **and** $B = A \vee * B$]
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus A \vee * B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ i$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma 1$ **and** $A = At\ i$ **and** $B = A \vee * B$]
by *auto*
from *eq2''* *eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ i \oplus At\ i \supset E$ **by** *simp*
from *eq1* **have** $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma 2 \oplus At\ i \oplus E \oplus A \Rightarrow C \downarrow j \wedge \Gamma 2 \oplus At\ i \oplus E \oplus B \Rightarrow C \downarrow k$
using *prems*(3)[**where** $\Gamma = \Gamma 2 \oplus At\ i \oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain $j\ k$ **where** *b1*: $j \leq n \wedge k \leq n$
and *b2*: $\Gamma 2 \oplus At\ i \oplus E \oplus A \Rightarrow C \downarrow j$
and *b3*: $\Gamma 2 \oplus At\ i \oplus E \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 2 \oplus At\ i \oplus At\ i \supset E \oplus A \Rightarrow C \downarrow j+1$ **using** *provable.ImpL0*[**where** $\Gamma = \Gamma 2 \oplus A$ **and** $i = i$ **and** $B = E$ **and** $C = C$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \Rightarrow C \downarrow j+1$ **using** *eq2* **by** *simp*
moreover
from *b3* **have** $\Gamma 2 \oplus At\ i \oplus At\ i \supset E \oplus B \Rightarrow C \downarrow k+1$ **using** *provable.ImpL0*[**where** $\Gamma = \Gamma 2 \oplus B$ **and** $i = i$ **and** $B = E$ **and** $C = C$ **and** $n = k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow C \downarrow k+1$ **using** *eq2* **by** *simp*

ultimately
show *?case* **using** *b1* **by** (*rule-tac x=j+1* **in** *exI*, *rule-tac x=k+1* **in** *exI*) (*auto*)
next
case (*ImpLC* Γ' D E C G n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \vee * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D \wedge * E) \supset C$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (D \wedge * E) \supset C$ **and** $B = A \vee * B$]
by *auto*
from *eq1* **have** $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma 1 \oplus D \supset (E \supset C) \oplus A \Rightarrow G \downarrow j \wedge \Gamma 1 \oplus D \supset (E \supset C) \oplus B \Rightarrow G \downarrow k$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus D \supset (E \supset C)$]
by (*auto simp only*:*union-ac*)
then obtain $j\ k$ **where** *b1*: $j \leq n \wedge k \leq n$
and *b2*: $\Gamma 1 \oplus D \supset (E \supset C) \oplus A \Rightarrow G \downarrow j$
and *b3*: $\Gamma 1 \oplus D \supset (E \supset C) \oplus B \Rightarrow G \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (D \wedge * E) \supset C \oplus A \Rightarrow G \downarrow j+1$ **using** *provable.ImpLC*[**where** $\Gamma = \Gamma 1 \oplus A$ **and** $A = D$ **and** $B = E$ **and** $C = C$ **and** $D = G$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \Rightarrow G \downarrow j+1$ **using** *eq2* **by** *simp*
moreover
from *b3* **have** $\Gamma 1 \oplus (D \wedge * E) \supset C \oplus B \Rightarrow G \downarrow k+1$ **using** *provable.ImpLC*[**where** $\Gamma = \Gamma 1 \oplus B$ **and** $A = D$ **and** $B = E$ **and** $C = C$ **and** $D = G$ **and** $n = k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow G \downarrow k+1$ **using** *eq2* **by** *simp*
ultimately
show *?case* **using** *b1* **by** (*rule-tac x=j+1* **in** *exI*,*rule-tac x=k+1* **in** *exI*) (*auto*)
next
case (*ImpLD* Γ' D C E G n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A \vee * B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D \vee * E) \supset C$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (D \vee * E) \supset C$ **and** $B = A \vee * B$]
by *auto*
from *eq1* **have** $\exists\ j\ k.\ j \leq n \wedge k \leq n \wedge \Gamma 1 \oplus D \supset C \oplus E \supset C \oplus A \Rightarrow G \downarrow j \wedge \Gamma 1 \oplus D \supset C \oplus E \supset C \oplus B \Rightarrow G \downarrow k$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus D \supset C \oplus E \supset C$]
by (*auto simp add*:*union-ac*)
then obtain $j\ k$ **where** *b1*: $j \leq n \wedge k \leq n$
and *b2*: $\Gamma 1 \oplus D \supset C \oplus E \supset C \oplus A \Rightarrow G \downarrow j$
and *b3*: $\Gamma 1 \oplus D \supset C \oplus E \supset C \oplus B \Rightarrow G \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (D \vee * E) \supset C \oplus A \Rightarrow G \downarrow j+1$ **using** *provable.ImpLD*[**where** $\Gamma = \Gamma 1 \oplus A$ **and** $A = D$ **and** $C = C$ **and** $B = E$ **and** $D = G$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \Rightarrow G \downarrow j+1$ **using** *eq2* **by** *simp*
moreover
from *b3* **have** $\Gamma 1 \oplus (D \vee * E) \supset C \oplus B \Rightarrow G \downarrow k+1$ **using** *provable.ImpLD*[**where** $\Gamma = \Gamma 1 \oplus B$

and $A{=}D$ **and** $C{=}C$ **and** $B{=}E$ **and** $D{=}G$ **and** $n{=}k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow G \downarrow k{+}1$ **using** *eq2* **by** *simp*
ultimately
show *?case* **using** *b1* **by** (*rule-tac* $x{=}j{+}1$ **in** *exI*,*rule-tac* $x{=}k{+}1$ **in** *exI*) (*auto*)
next
case (*ImpLL* Γ' D E C n G m Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus A{\vee}{*}B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (D{\supset}E){\supset}C$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}(D{\supset}E){\supset}C$ **and** $B{=}A{\vee}{*}B$] **by** *auto*
from *eq1* **have** $\exists\ j\ k.\ j{\leq}n \wedge k{\leq}n \wedge \Gamma 1 \oplus D \oplus E{\supset}C \oplus A \Rightarrow E \downarrow j \wedge \Gamma 1 \oplus D \oplus E{\supset}C \oplus B \Rightarrow E \downarrow k$ **using** *prems*(3)[**where** $\Gamma{=}\Gamma 1 \oplus D \oplus E{\supset}C$]
by (*auto simp add*:*union-ac*)
then obtain $j\ k$ **where** *b1*: $j{\leq}n \wedge k{\leq}n$
and *b2*: $\Gamma 1 \oplus D \oplus E{\supset}C \oplus A \Rightarrow E \downarrow j$
and *b3*: $\Gamma 1 \oplus D \oplus E{\supset}C \oplus B \Rightarrow E \downarrow k$ **by** *blast*
moreover
from *eq1* **have** $\exists\ j'\ k'.\ j'{\leq}m \wedge k'{\leq}m \wedge \Gamma 1{\oplus}C \oplus A \Rightarrow G \downarrow j' \wedge \Gamma 1{\oplus}C \oplus B \Rightarrow G \downarrow k'$ **using** *prems*(5)[**where** $\Gamma{=}\Gamma 1 \oplus C$]
by (*auto simp add*:*union-ac*)
then obtain $j'\ k'$ **where** *c1*: $j'{\leq}m \wedge k'{\leq}m$
and *c2*: $\Gamma 1{\oplus}C \oplus A \Rightarrow G \downarrow j'$
and *c3*: $\Gamma 1{\oplus}C \oplus B \Rightarrow G \downarrow k'$ **by** *auto*
ultimately
have $\Gamma 1 \oplus (D{\supset}E){\supset}C \oplus A \Rightarrow G \downarrow j{+}j'{+}1$ **using** *provable.ImpLL*[**where** $\Gamma{=}\Gamma 1{\oplus}A$ **and** $A{=}D$ **and** $B{=}E$ **and** $C{=}C$ **and** $D{=}G$ **and** $n{=}j$ **and** $m{=}j'$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus A \Rightarrow G \downarrow j{+}j'{+}1$ **using** *eq2* **by** *simp*
moreover
from *b3 c3* **have** $\Gamma 1 \oplus (D{\supset}E){\supset}C \oplus B \Rightarrow G \downarrow k{+}k'{+}1$ **using** *provable.ImpLL*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}D$ **and** $B{=}E$ **and** $C{=}C$ **and** $D{=}G$ **and** $n{=}k$ **and** $m{=}k'$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow G \downarrow k{+}k'{+}1$ **using** *eq2* **by** *simp*
ultimately
show *?case* **using** *b1 c1* **by** (*rule-tac* $x{=}j{+}j'{+}1$ **in** *exI*,*rule-tac* $x{=}k{+}k'{+}1$ **in** *exI*) (*auto*)
next
case (*DisjL* Γ' E C n F m Γ'')
have $E{\vee}{*}F = A{\vee}{*}B \vee E{\vee}{*}F \neq A{\vee}{*}B$ **by** *blast*
moreover
{assume $E{\vee}{*}F = A{\vee}{*}B$
from *prems* **have** $\Gamma' = \Gamma'' \wedge E{=}A \wedge F{=}B$ **by** *auto*
then have $\Gamma'' \oplus A \Rightarrow C \downarrow n$ **and** $\Gamma'' \oplus B \Rightarrow C \downarrow m$ **using** *prems* **by** *auto*
then have $\exists\ j\ k.\ j{\leq}n{+}m{+}1 \wedge k{\leq}n{+}m{+}1 \wedge \Gamma'' \oplus A \Rightarrow C \downarrow j \wedge \Gamma'' \oplus B \Rightarrow C \downarrow k$ **apply**

```
(rule-tac x=n in exI,rule-tac x=m in exI)
      by auto
  }
  moreover
  {assume E∨*F ≠ A∨*B
  then obtain Γ1 where eq1: Γ' = Γ1 ⊕ A∨*B
                  and eq2: Γ'' = Γ1 ⊕ E∨*F
      using midMultiset[where Γ=Γ' and Γ'=Γ'' and A=E∨*F and B=A∨*B] prems by auto
  from eq1 prems(3)[where Γ=Γ1⊕E] have ∃ j1 k1. j1≤n ∧ k1≤n ∧ Γ1 ⊕ E ⊕ A ⇒ C ↓ j1 ∧
Γ1 ⊕ E ⊕ B ⇒ C ↓ k1
       by (auto simp add:union-ac)
  then obtain j1 k1 where a: j1≤n ∧ k1≤n
                  and a1: Γ1 ⊕ E ⊕ A ⇒ C ↓ j1
                  and a2: Γ1 ⊕ E ⊕ B ⇒ C ↓ k1 by blast
  from eq1 prems(5)[where Γ=Γ1⊕F] have ∃ j2 k2. j2≤m ∧ k2≤m ∧ Γ1 ⊕ F ⊕ A ⇒ C ↓ j2
∧ Γ1 ⊕ F ⊕ B ⇒ C ↓ k2
       by (auto simp add:union-ac)
  then obtain j2 k2 where b: j2≤m ∧ k2≤m
                  and b1: Γ1 ⊕ F ⊕ A ⇒ C ↓ j2
                  and b2: Γ1 ⊕ F ⊕ B ⇒ C ↓ k2 by blast
  from a1 b1 have Γ1 ⊕ E∨*F ⊕ A ⇒ C ↓ j1+j2+1 using provable.DisjL[where Γ=Γ1⊕A
and A=E and B=F]
      by (auto simp add:union-ac)
  moreover
  from a2 b2 have Γ1 ⊕ E∨*F ⊕ B ⇒ C ↓ k1+k2+1 using provable.DisjL[where Γ=Γ1⊕B
and A=E and B=F]
      by (auto simp add:union-ac)
  ultimately
  have ∃ j k. j≤n+m+1 ∧ k≤n+m+1 ∧ Γ'' ⊕ A ⇒ C ↓ j ∧ Γ'' ⊕ B ⇒ C ↓ k using eq2 a b
    apply (rule-tac x=j1+j2+1 in exI, rule-tac x=k1+k2+1 in exI) by auto
  }
  ultimately
  show ?case by blast
qed

lemma inversionImpL0:
  assumes Γ ⊕ At i ⊃ B ⇒ C ↓ n
  shows ∃ j≤n. Γ ⊕ B ⇒ C ↓ j
  using assms
proof (induct Γ≡Γ ⊕ At i ⊃ B C n arbitrary:Γ)
  case (Ax i' Γ')
  then have At i' :# Γ by auto
  then have Γ ⊕ B ⇒ At i' ↓ 0 by auto
```

then show *?case* **by** *blast*
next
case (*LBot* Γ' C)
then have *ff* :# Γ **by** *auto*
then have $\Gamma \oplus B \Rightarrow C \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*ImpR* Γ' E F k)
then have $\Gamma' \oplus E = \Gamma \oplus At\ i \supset B \oplus E$ **by** *auto*
then have $\exists\ j.\ j \leq k \wedge \Gamma \oplus B \oplus E \Rightarrow F \downarrow j$ **using** *prems*(*3*)[**where** Γ=$\Gamma \oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *c1*: $j \leq k$
and *c2*: $\Gamma \oplus B \oplus E \Rightarrow F \downarrow j$ **by** *auto*
from *c2* **have** $\Gamma \oplus B \Rightarrow E \supset F \downarrow j+1$ **using** *provable.ImpR*[**where** Γ=$\Gamma \oplus B$ **and** A=E **and** B=F] **by** *auto*
then show *?case* **using** *c1* **by** *auto*
next
case (*ConjR* Γ' E k F l)
then have $\exists j \leq k.\ \Gamma \oplus B \Rightarrow E \downarrow j$ **and** $\exists j \leq l.\ \Gamma \oplus B \Rightarrow F \downarrow j$ **by** *auto*
then obtain *j1 j2* **where** *c1*: $j1 \leq k$
and *c2*: $\Gamma \oplus B \Rightarrow E \downarrow j1$
and *c3*: $j2 \leq l$
and *c4*: $\Gamma \oplus B \Rightarrow F \downarrow j2$ **by** *auto*
then show *?case* **using** *provable.ConjR*[**where** Γ=$\Gamma \oplus B$ **and** n=*j1* **and** m=*j2* **and** A=E **and** B=F]
apply (*rule-tac* x=*j1*+*j2*+*1* **in** *exI*) **by** *auto*
next
case (*ConjL* Γ' E F C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus At\ i \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \wedge * F$ **using** *midMultiset*[**where** Γ=Γ' **and** A=$E \wedge * F$ **and** Γ'=Γ'' **and** B=$At\ i \supset B$] **by** *auto*
from *eq1 prems*(*3*)[**where** Γ=$\Gamma 1 \oplus E \oplus F$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
then obtain j **where** *eq3*: $j \leq n$ **and** $\Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow j$ **by** *blast*
then have $\Gamma 1 \oplus E \wedge * F \oplus B \Rightarrow C \downarrow j+1$ **using** *provable.ConjL*[**where** Γ=$\Gamma 1 \oplus B$ **and** A=E **and** B=F] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2 eq3* **by** *auto*
next
case (*DisjR1* Γ' E n F)
then have $\exists j \leq n.\ \Gamma \oplus B \Rightarrow E \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j \leq n$ **and** $\Gamma \oplus B \Rightarrow E \downarrow j$ **by** *blast*
then have $\Gamma \oplus B \Rightarrow E \vee * F \downarrow j+1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*

next
case (*DisjR2* Γ' F n E)
then have $\exists j{\leq}n.\ \Gamma \oplus B \Rightarrow F \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j{\leq}n$ **and** $\Gamma \oplus B \Rightarrow F \downarrow j$ **by** *blast*
then have $\Gamma \oplus B \Rightarrow E{\vee}{*}F \downarrow j{+}1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjL* Γ' E C n F m Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus At\ i \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E{\vee}{*}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}E{\vee}{*}F$ **and** $B{=}At\ i \supset B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E$] **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
moreover
from *eq1 prems*(*5*)[**where** $\Gamma{=}\Gamma 1{\oplus}F$] **have** $\exists\ k{\leq}m.\ \Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
ultimately
obtain j k **where** *a*: $j{\leq}n \wedge k{\leq}m$
and *b*: $\Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow j$
and *c*: $\Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b c* **have** $\Gamma 1 \oplus E{\vee}{*}F \oplus B \Rightarrow C \downarrow j{+}k{+}1$ **using** *provable.DisjL*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a eq2* **apply** (*rule-tac* $x{=}j{+}k{+}1$ **in** *exI*) **by** *auto*
next
case (*ImpLC* Γ' E F D C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus At\ i \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E{\wedge}{*}F){\supset}D$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}(E{\wedge}{*}F){\supset}D$ **and** $B{=}At\ i \supset B$] **by** *auto*
from *eq1* **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \supset (F{\supset}D) \oplus B \Rightarrow C \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\supset}(F{\supset}D)$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus E \supset (F{\supset}D) \oplus B \Rightarrow C \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (E{\wedge}{*}F){\supset}D \oplus B \Rightarrow C \downarrow j{+}1$ **using** *provable.ImpLC*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}D$ **and** $D{=}C$ **and** $n{=}j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma''{\oplus}B \Rightarrow C \downarrow j{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x{=}j{+}1$ **in** *exI*) (*auto*)
next
case (*ImpLD* Γ' E D F C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus At\ i \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E{\vee}{*}F){\supset}D$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}(E{\vee}{*}F){\supset}D$ **and** $B{=}At\ i \supset B$] **by** *auto*

from *eq1* **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E{\supset}D \oplus F{\supset}D \oplus B \Rightarrow C \downarrow j$ **using** *prems*(3)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\supset}D{\oplus}F{\supset}D$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus E{\supset}D \oplus F{\supset}D \oplus B \Rightarrow C \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (E{\vee}{*}F){\supset}D \oplus B \Rightarrow C \downarrow j{+}1$ **using** *provable.ImpLD*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}D$ **and** $D{=}C$ **and** $n{=}j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma''{\oplus}B \Rightarrow C \downarrow j{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x{=}j{+}1$ **in** *exI*) (*auto*)
next
case (*ImpLL* $\Gamma'\ E\ F\ D\ n\ C\ m\ \ \Gamma''$)
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus At\ i \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E{\supset}F){\supset}D$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}(E{\supset}F){\supset}D$ **and** $B{=}At\ i \supset B$] **by** *auto*
from *eq1* **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \oplus F{\supset}D \oplus B \Rightarrow F \downarrow j$ **using** *prems*(3)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\oplus}F{\supset}D$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus E \oplus F{\supset}D \oplus B \Rightarrow F \downarrow j$ **by** *blast*
moreover
from *eq1* **have** $\exists\ \ k{\leq}m.\ \Gamma 1 \oplus D \oplus B \Rightarrow C \downarrow k$ **using** *prems*(5)[**where** $\Gamma{=}\Gamma 1{\oplus}D$] **by** (*auto simp add*:*union-ac*)
then obtain *k* **where** *c1*: $k{\leq}m$
and *c2*: $\Gamma 1 \oplus D \oplus B \Rightarrow C \downarrow k$ **by** *blast*
ultimately
have $\Gamma 1 \oplus (E{\supset}F){\supset}D \oplus B \Rightarrow C \downarrow j{+}k{+}1$ **using** *provable.ImpLL*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}D$ **and** $D{=}C$ **and** $n{=}j$ **and** $m{=}k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow C \downarrow j{+}k{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1 c1* **by** (*rule-tac* $x{=}j{+}k{+}1$ **in** *exI*) (*auto*)
next
case (*ImpL0* $\Gamma'\ j\ D\ C\ n\ \Gamma''$)
have $(At\ i \supset B = At\ j \supset D) \vee (At\ i \supset B \neq At\ j \supset D)$ **by** *blast*
moreover
{**assume** $At\ i \supset B = At\ j \supset D$
then have $B{=}D$ **by** *auto*
from *prems*(3) **have** $\Gamma' \oplus At\ j = \Gamma''$ **using** *prems* **by** *auto*
then have $\Gamma'' \oplus B \Rightarrow C \downarrow n$ **using** ‹$B{=}D$› **by** *auto*
then have $\exists\ j{\leq}n{+}1.\ \Gamma'' \oplus B \Rightarrow C \downarrow j$ **by** (*rule-tac* $x{=}n$ **in** *exI*) (*auto*)
}
moreover
{**assume** $At\ i \supset B \neq At\ j \supset D$
then obtain $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ j = \Gamma 1 \oplus At\ i \supset B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ j \supset D$

using *midMultiset*[**where** $\Gamma{=}\Gamma' \oplus At\ j$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}At\ j \supset D$ **and** $B{=}At\ i \supset B$] *prems*
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus At\ i \supset B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ j$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma 1$ **and** $A{=}At\ j$ **and** $B{=}At\ i \supset B$]
by *auto*
from *eq2'' eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ j \oplus At\ j \supset D$ **by** *simp*
from *eq1* **have** $\exists\ k{\leq}n.\ \Gamma 2 \oplus At\ j \oplus D \oplus B \Rightarrow C \downarrow k$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 2 \oplus At\ j \oplus D$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *b1*: $k{\leq}n$
and *b2*: $\Gamma 2 \oplus At\ j \oplus D \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 2 \oplus At\ j \oplus At\ j \supset D \oplus B \Rightarrow C \downarrow k{+}1$ **using** *provable.ImpL0*[**where** $\Gamma{=}\Gamma 2 \oplus B$ **and** $i{=}j$ **and** $B{=}D$ **and** $C{=}C$ **and** $n{=}k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow C \downarrow k{+}1$ **using** *eq2* **by** *simp*
then have $\exists\ j{\leq}n{+}1.\ \Gamma''{\oplus}B \Rightarrow C \downarrow j$ **using** *b1* **by** (*rule-tac* $x{=}k{+}1$ **in** *exI*) (*auto*)
}
ultimately
show *?case* **by** *blast*
qed

lemma *inversionImpLC*:
assumes $\Gamma \oplus (S{\wedge}{*}T) \supset B \Rightarrow C \downarrow n$
shows $\exists\ j{\leq}n.\ \Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$
using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma \oplus (S{\wedge}{*}T) \supset B\ C\ n$ *arbitrary*:Γ)
case ($Ax\ i'\ \Gamma'$)
then have $At\ i' :\#\ \Gamma$ **by** *auto*
then have $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow At\ i' \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case ($LBot\ \Gamma'\ C$)
then have $ff :\#\ \Gamma$ **by** *auto*
then have $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case ($ImpR\ \Gamma'\ E\ F\ k$)
then have $\Gamma' \oplus E = \Gamma \oplus (S{\wedge}{*}T) \supset B \oplus E$ **by** *auto*
then have $\exists\ j.\ j{\leq}k \wedge \Gamma \oplus S{\supset}(T{\supset}B) \oplus E \Rightarrow F \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma{\oplus}E$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *c1*: $j{\leq}k$
and *c2*: $\Gamma \oplus S{\supset}(T{\supset}B) \oplus E \Rightarrow F \downarrow j$ **by** *auto*

from *c2* **have** $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E{\supset}F \downarrow j{+}1$ **using** *provable.ImpR*[**where** $\Gamma{=}\Gamma \oplus S{\supset}(T{\supset}B)$ **and** $A{=}E$ **and** $B{=}F$] **by** *auto*
then show *?case* **using** *c1* **by** *auto*
next
case (*ConjR* Γ' *E k F l*)
then have $\exists j{\leq}k.\ \Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E \downarrow j$ **and** $\exists j{\leq}l.\ \Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow F \downarrow j$ **by** *auto*
then obtain *j1 j2* **where** *c1*: $j1 \leq k$
and *c2*: $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E \downarrow j1$
and *c3*: $j2 \leq l$
and *c4*: $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow F \downarrow j2$ **by** *auto*
then show *?case* **using** *provable.ConjR*[**where** $\Gamma{=}\Gamma{\oplus}S{\supset}(T{\supset}B)$ **and** $n{=}j1$ **and** $m{=}j2$ **and** $A{=}E$ **and** $B{=}F$]
apply (*rule-tac x=j1+j2+1* **in** *exI*) **by** *auto*
next
case (*ConjL* Γ' *E F C n* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S{\wedge}{*}T){\supset}\ B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E{\wedge}{*}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $A{=}E{\wedge}{*}F$ **and** $\Gamma'{=}\Gamma''$ **and** $B{=}(S{\wedge}{*}T){\supset}\ B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\oplus}F$] **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \oplus F \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *eq3*: $j{\leq}n$ **and** $\Gamma 1 \oplus E \oplus F \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **by** *blast*
then have $\Gamma 1 \oplus E{\wedge}{*}F \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j{+}1$ **using** *provable.ConjL*[**where** $\Gamma{=}\Gamma 1{\oplus}S{\supset}(T{\supset}B)$ **and** $A{=}E$ **and** $B{=}F$] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2 eq3* **by** *auto*
next
case (*DisjR1* Γ' *E n F*)
then have $\exists j{\leq}n.\ \Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E \downarrow j$ **by** *auto*
then obtain *j* **where** *eq*:$j{\leq}n$ **and** $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E \downarrow j$ **by** *blast*
then have $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E{\vee}{*}F \downarrow j{+}1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ' *F n E*)
then have $\exists j{\leq}n.\ \Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow F \downarrow j$ **by** *auto*
then obtain *j* **where** *eq*:$j{\leq}n$ **and** $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow F \downarrow j$ **by** *blast*
then have $\Gamma \oplus S{\supset}(T{\supset}B) \Rightarrow E{\vee}{*}F \downarrow j{+}1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjL* Γ' *E C n F m* Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S{\wedge}{*}T){\supset}\ B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E{\vee}{*}F$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}E{\vee}{*}F$ **and** $B{=}(S{\wedge}{*}T){\supset}\ B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E$] **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)

moreover
from *eq1 prems*(*5*)[**where** Γ=Γ*1*⊕*F*] **have** ∃ *k*≤*m*. Γ*1* ⊕ *F* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *k* **by** (*auto simp add*:*union-ac*)
ultimately
obtain *j k* **where** *a*: *j*≤*n* ∧ *k*≤*m*
and *b*: Γ*1* ⊕ *E* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j*
and *c*: Γ*1* ⊕ *F* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *k* **by** *blast*
from *b c* **have** Γ*1* ⊕ *E*∨∗*F* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j*+*k*+*1* **using** *provable.DisjL*[**where** Γ=Γ*1*⊕*S*⊃(*T*⊃*B*) **and** *A*=*E* **and** *B*=*F*]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a eq2* **apply** (*rule-tac x*=*j*+*k*+*1* **in** *exI*) **by** *auto*
next
case (*ImpLD* Γ′ *E D F C n* Γ′′)
then obtain Γ*1* **where** *eq1*: Γ′ = Γ*1* ⊕ (*S*∧∗*T*) ⊃ *B*
and *eq2*: Γ′′ = Γ*1* ⊕ (*E*∨∗*F*)⊃*D* **using** *midMultiset*[**where** Γ=Γ′ **and** Γ′=Γ′′ **and** *A*=(*E*∨∗*F*)⊃*D* **and** *B*=(*S*∧∗*T*) ⊃ *B*] **by** *auto*
from *eq1* **have** ∃ *j*≤*n*. Γ*1* ⊕ *E*⊃*D* ⊕ *F*⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j* **using** *prems*(*3*)[**where** Γ=Γ*1*⊕*E*⊃*D*⊕*F*⊃*D*] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: *j*≤*n*
and *b2*: Γ*1* ⊕ *E*⊃*D* ⊕ *F*⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j* **by** *blast*
from *b2* **have** Γ*1* ⊕ (*E*∨∗*F*)⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j*+*1* **using** *provable.ImpLD*[**where** Γ=Γ*1*⊕*S*⊃(*T*⊃*B*) **and** *A*=*E* **and** *B*=*F* **and** *C*=*D* **and** *D*=*C* **and** *n*=*j*]
by (*auto simp add*:*union-ac*)
then have Γ′′⊕*S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j*+*1* **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac x*=*j*+*1* **in** *exI*) (*auto*)
next
case (*ImpLL* Γ′ *E F D n C m* Γ′′)
then obtain Γ*1* **where** *eq1*: Γ′ = Γ*1* ⊕ (*S*∧∗*T*) ⊃ *B*
and *eq2*: Γ′′ = Γ*1* ⊕ (*E*⊃*F*)⊃*D* **using** *midMultiset*[**where** Γ=Γ′ **and** Γ′=Γ′′ **and** *A*=(*E*⊃*F*)⊃*D* **and** *B*=(*S*∧∗*T*) ⊃ *B*] **by** *auto*
from *eq1* **have** ∃ *j*≤*n*. Γ*1* ⊕ *E* ⊕ *F*⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *F* ↓ *j* **using** *prems*(*3*)[**where** Γ=Γ*1*⊕*E*⊕*F*⊃*D*] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *b1*: *j*≤*n*
and *b2*: Γ*1* ⊕ *E* ⊕ *F*⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *F* ↓ *j* **by** *blast*
moreover
from *eq1* **have** ∃ *k*≤*m*. Γ*1* ⊕ *D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *k* **using** *prems*(*5*)[**where** Γ=Γ*1*⊕*D*] **by** (*auto simp add*:*union-ac*)
then obtain *k* **where** *c1*: *k*≤*m*
and *c2*: Γ*1* ⊕ *D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *k* **by** *blast*
ultimately
have Γ*1* ⊕ (*E*⊃*F*)⊃*D* ⊕ *S*⊃(*T*⊃*B*) ⇒ *C* ↓ *j*+*k*+*1* **using** *provable.ImpLL*[**where** Γ=Γ*1*⊕*S*⊃(*T*⊃*B*) **and** *A*=*E* **and** *B*=*F* **and** *C*=*D* **and** *D*=*C* **and** *n*=*j* **and** *m*=*k*]
by (*auto simp add*:*union-ac*)

then have $\Gamma'' \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j{+}k{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1 c1* **by** (*rule-tac x*=$j{+}k{+}1$ **in** *exI*) (*auto*)
next
case (*ImplL0* Γ' j D C n Γ'')
then obtain $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ j = \Gamma 1 \oplus (S{\wedge}{*}T) \supset B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ j \supset D$
using *midMultiset*[**where** Γ=$\Gamma' \oplus At\ j$ **and** Γ'=Γ'' **and** A=$At\ j \supset D$ **and** B=$(S{\wedge}{*}T) \supset B$] *prems*
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus (S{\wedge}{*}T){\supset}\ B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ j$ **using** *midMultiset*[**where** Γ=Γ' **and** Γ'=$\Gamma 1$ **and** A=$At\ j$ **and** B=$(S{\wedge}{*}T) \supset B$]
by *auto*
from *eq2''* *eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ j \oplus At\ j \supset D$ **by** *simp*
from *eq1* **have** $\exists\ k{\leq}n.\ \Gamma 2 \oplus At\ j \oplus D \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow k$ **using** *prems*(*3*)[**where** Γ=$\Gamma 2{\oplus}At\ j \oplus D$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *b1*: $k{\leq}n$
and *b2*: $\Gamma 2 \oplus At\ j \oplus D \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 2 \oplus At\ j \oplus At\ j \supset D \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow k{+}1$ **using** *provable.ImplL0*[**where** Γ=$\Gamma 2{\oplus}S{\supset}(T{\supset}B)$ **and** i=j **and** B=D **and** C=C **and** n=k]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow k{+}1$ **using** *eq2* **by** *simp*
then show $\exists\ j{\leq}n{+}1.\ \Gamma''{\oplus}S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **using** *b1* **by** (*rule-tac x*=$k{+}1$ **in** *exI*) (*auto*)
next
case (*ImplLC* Γ' E F D C n Γ'')
have $((E{\wedge}{*}F){\supset}D = (S{\wedge}{*}T){\supset}B) \vee ((E{\wedge}{*}F){\supset}D \neq (S{\wedge}{*}T){\supset}B)$ **by** *blast*
moreover
{assume $(E{\wedge}{*}F){\supset}D = (S{\wedge}{*}T){\supset}B$
then have E=$S \wedge F$=$T \wedge D$=$B \wedge \Gamma'$=Γ'' **using** *prems* **by** *auto*
then have $\Gamma'' \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow n$ **using** *prems* **by** *simp*
then have $\exists\ j{\leq}n{+}1.\ \Gamma'' \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **by** (*rule-tac x*=n **in** *exI*) (*auto*)
}
moreover
{assume $(E{\wedge}{*}F){\supset}D \neq (S{\wedge}{*}T){\supset}B$
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S{\wedge}{*}T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E{\wedge}{*}F){\supset}D$
using *midMultiset*[**where** Γ=Γ' **and** Γ'=Γ'' **and** A=$(E{\wedge}{*}F){\supset}D$ **and** B=$(S{\wedge}{*}T) \supset B$] *prems* **by** *auto*
from *eq1* **have** $\exists\ j{\leq}n.\ \Gamma 1 \oplus E \supset (F{\supset}D) \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **using** *prems*(*3*)[**where** Γ=$\Gamma 1{\oplus}E{\supset}(F{\supset}D)$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus E \supset (F{\supset}D) \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (E{\wedge}{*}F){\supset}D \oplus S{\supset}(T{\supset}B) \Rightarrow C \downarrow j{+}1$ **using** *provable.ImplLC*[**where**

Γ=Γ*1*$\oplus S\supset(T\supset B)$ **and** A=E **and** B=F **and** C=D **and** D=C **and** n=j]
by (*auto simp add*:*union-ac*)
then have $\Gamma''\oplus S\supset(T\supset B) \Rightarrow C \downarrow j{+}1$ **using** *eq2* **by** *simp*
then have $\exists\ j{\leq}n{+}1.\ \Gamma''\oplus S\supset(T\supset B) \Rightarrow C \downarrow j$ **using** *b1* **by** (*rule-tac x*=$j{+}1$ **in** *exI*) (*auto*)
}
ultimately
show *?case* **by** *blast*
qed

lemma *inversionImpLD*:
assumes $\Gamma \oplus (S{\vee}{*}T) \supset B \Rightarrow C \downarrow n$
shows $\exists\ j{\leq}n.\ \Gamma \oplus S\supset B \oplus T\supset B \Rightarrow C \downarrow j$
using *assms*
proof (*induct* $\Gamma{\equiv}\Gamma \oplus (S{\vee}{*}T) \supset B\ C\ n$ *arbitrary*:Γ)
case (*Ax* $i'\ \Gamma'$)
then have *At* i' :# Γ **by** *auto*
then have $\Gamma \oplus S\supset B \oplus T\supset B \Rightarrow At\ i' \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*LBot* $\Gamma'\ C$)
then have *ff* :# Γ **by** *auto*
then have $\Gamma \oplus S\supset B \oplus T\supset B \Rightarrow C \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*ImpR* $\Gamma'\ E\ F\ k$)
then have $\Gamma' \oplus E = \Gamma \oplus (S{\vee}{*}T) \supset B \oplus E$ **by** *auto*
then have $\exists\ j.\ j{\leq}k \wedge \Gamma \oplus S\supset B \oplus T\supset B \oplus E \Rightarrow F \downarrow j$ **using** *prems*(*3*)[**where** Γ=$\Gamma\oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *c1*: $j{\leq}k$
and *c2*: $\Gamma \oplus S\supset B \oplus T\supset B \oplus E \Rightarrow F \downarrow j$ **by** *auto*
from *c2* **have** $\Gamma \oplus S\supset B\oplus(T\supset B) \Rightarrow E\supset F \downarrow j{+}1$ **using** *provable.ImpR*[**where** Γ=$\Gamma \oplus S\supset B\oplus(T\supset B)$ **and** A=E **and** B=F] **by** *auto*
then show *?case* **using** *c1* **by** *auto*
next
case (*ConjR* $\Gamma'\ E\ k\ F\ l$)
then have $\exists j{\leq}k.\ \Gamma \oplus S\supset B\oplus(T\supset B) \Rightarrow E \downarrow j$ **and** $\exists j{\leq}l.\ \Gamma \oplus S\supset B\oplus(T\supset B) \Rightarrow F \downarrow j$ **by** *auto*
then obtain *j1 j2* **where** *c1*: $j1 \leq k$
and *c2*: $\Gamma \oplus S\supset B\oplus(T\supset B) \Rightarrow E \downarrow j1$
and *c3*: $j2 \leq l$
and *c4*: $\Gamma \oplus S\supset B\oplus(T\supset B) \Rightarrow F \downarrow j2$ **by** *auto*
then show *?case* **using** *provable.ConjR*[**where** Γ=$\Gamma\oplus S\supset B\oplus(T\supset B)$ **and** n=$j1$ **and** m=$j2$ **and** A=E **and** B=F]
apply (*rule-tac x*=$j1{+}j2{+}1$ **in** *exI*) **by** *auto*

next
case (*ConjL* Γ' E F C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \vee * T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \wedge * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $A = E \wedge * F$ **and** $\Gamma' = \Gamma''$ **and** $B = (S \vee * T) \supset B$] **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E \oplus F$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
then obtain j **where** *eq3*: $j \leq n$ **and** $\Gamma 1 \oplus E \oplus F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j$ **by** *blast*
then have $\Gamma 1 \oplus E \wedge * F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j+1$ **using** *provable.ConjL*[**where** $\Gamma = \Gamma 1 \oplus S \supset B \oplus (T \supset B)$ **and** $A = E$ **and** $B = F$] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2 eq3* **by** *auto*
next
case (*DisjR1* Γ' E n F)
then have $\exists j \leq n.\ \Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow E \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j \leq n$ **and** $\Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow E \downarrow j$ **by** *blast*
then have $\Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow E \vee * F \downarrow j+1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ' F n E)
then have $\exists j \leq n.\ \Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow F \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j \leq n$ **and** $\Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow F \downarrow j$ **by** *blast*
then have $\Gamma \oplus S \supset B \oplus (T \supset B) \Rightarrow E \vee * F \downarrow j+1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjL* Γ' E C n F m Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \vee * T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \vee * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = E \vee * F$ **and** $B = (S \vee * T) \supset B$] **by** *auto*
from *eq1 prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
moreover
from *eq1 prems*(5)[**where** $\Gamma = \Gamma 1 \oplus F$] **have** $\exists\ k \leq m.\ \Gamma 1 \oplus F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
ultimately
obtain j k **where** a: $j \leq n \wedge k \leq m$
and b: $\Gamma 1 \oplus E \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j$
and c: $\Gamma 1 \oplus F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **by** *blast*
from b c **have** $\Gamma 1 \oplus E \vee * F \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j+k+1$ **using** *provable.DisjL*[**where** $\Gamma = \Gamma 1 \oplus S \supset B \oplus (T \supset B)$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** a *eq2* **apply** (*rule-tac* $x = j+k+1$ **in** *exI*) **by** *auto*
next
case (*ImpLL* Γ' E F D n C m Γ'')

then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \vee * T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E \supset F) \supset D$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (E \supset F) \supset D$ **and** $B = (S \vee * T) \supset B$] **by** *auto*
from *eq1* **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus F \supset D \oplus S \supset B \oplus (T \supset B) \Rightarrow F \downarrow j$ **using** *prems*(3)[**where** $\Gamma = \Gamma 1 \oplus E \oplus F \supset D$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus E \oplus F \supset D \oplus S \supset B \oplus (T \supset B) \Rightarrow F \downarrow j$ **by** *blast*
moreover
from *eq1* **have** $\exists\ k \leq m.\ \Gamma 1 \oplus D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **using** *prems*(5)[**where** $\Gamma = \Gamma 1 \oplus D$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *c1*: $k \leq m$
and *c2*: $\Gamma 1 \oplus D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **by** *blast*
ultimately
have $\Gamma 1 \oplus (E \supset F) \supset D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j+k+1$
using *provable.ImpLL*[**where** $\Gamma = \Gamma 1 \oplus S \supset B \oplus (T \supset B)$ **and** $A = E$ **and** $B = F$ **and** $C = D$ **and** $D = C$ **and** $n = j$ **and** $m = k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j+k+1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1 c1* **by** (*rule-tac* $x = j+k+1$ **in** *exI*) (*auto*)
next
case (*ImpL0* $\Gamma'\ j\ D\ C\ n\ \Gamma''$)
then obtain $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ j = \Gamma 1 \oplus (S \vee * T) \supset B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ j \supset D$
using *midMultiset*[**where** $\Gamma = \Gamma' \oplus At\ j$ **and** $\Gamma' = \Gamma''$ **and** $A = At\ j \supset D$ **and** $B = (S \vee * T) \supset B$] *prems*
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus (S \vee * T) \supset B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ j$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma 1$ **and** $A = At\ j$ **and** $B = (S \vee * T) \supset B$]
by *auto*
from *eq2''* *eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ j \oplus At\ j \supset D$ **by** *simp*
from *eq1* **have** $\exists\ k \leq n.\ \Gamma 2 \oplus At\ j \oplus D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **using** *prems*(3)[**where** $\Gamma = \Gamma 2 \oplus At\ j \oplus D$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *b1*: $k \leq n$
and *b2*: $\Gamma 2 \oplus At\ j \oplus D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 2 \oplus At\ j \oplus At\ j \supset D \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k+1$ **using** *provable.ImpL0*[**where** $\Gamma = \Gamma 2 \oplus S \supset B \oplus (T \supset B)$ **and** $i = j$ **and** $B = D$ **and** $C = C$ **and** $n = k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow k+1$ **using** *eq2* **by** *simp*
then show $\exists\ j \leq n+1.\ \Gamma'' \oplus S \supset B \oplus (T \supset B) \Rightarrow C \downarrow j$ **using** *b1* **by** (*rule-tac* $x = k+1$ **in** *exI*) (*auto*)
next
case (*ImpLC* $\Gamma'\ E\ F\ D\ C\ n\ \Gamma''$)
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \vee * T) \supset B$

```
                and eq2: Γ'' = Γ1 ⊕ (E∧∗F)⊃D
      using midMultiset[where Γ=Γ' and Γ'=Γ'' and A=(E∧∗F)⊃D and B=(S∨∗T) ⊃ B]
prems by auto
  from eq1 have ∃ j≤n. Γ1 ⊕ E ⊃ (F⊃D) ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j using prems(3)[where
Γ=Γ1⊕E⊃(F⊃D)] by (auto simp add:union-ac)
  then obtain j where b1: j≤n
               and b2: Γ1 ⊕ E ⊃ (F⊃D) ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j by blast
  from b2 have Γ1 ⊕ (E∧∗F)⊃D ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j+1
      using provable.ImpLC[where Γ=Γ1⊕S⊃B⊕(T⊃B) and A=E and B=F and C=D and
D=C and n=j]
      by (auto simp add:union-ac)
  then have Γ''⊕S⊃B⊕(T⊃B) ⇒ C ↓ j+1 using eq2 by simp
  then show ?case using b1 by (rule-tac x=j+1 in exI) (auto)
next
  case (ImpLD Γ' E D F C n Γ'')
  have ((E∨∗F)⊃D = (S∨∗T)⊃B) ∨ ((E∨∗F)⊃D ≠ (S∨∗T)⊃B) by blast
   moreover
   {assume (E∨∗F)⊃D = (S∨∗T)⊃B
    then have E=S ∧ F=T ∧ D=B ∧ Γ'=Γ'' using prems by auto
    then have Γ'' ⊕ S⊃B ⊕ T⊃B ⇒ C ↓ n using prems by auto
    then have ∃ j≤n+1. Γ'' ⊕ S⊃B ⊕ T⊃B ⇒ C ↓ j by (rule-tac x=n in exI) (auto)
   }
   moreover
   {assume (E∨∗F)⊃D ≠ (S∨∗T)⊃B
    then obtain Γ1 where eq1: Γ' = Γ1 ⊕ (S∨∗T) ⊃ B
                 and eq2: Γ'' = Γ1 ⊕ (E∨∗F)⊃D
       using midMultiset[where Γ=Γ' and Γ'=Γ'' and A=(E∨∗F)⊃D and B=(S∨∗T) ⊃ B]
prems by auto
    from eq1 have ∃ j≤n. Γ1 ⊕ E⊃D ⊕ F⊃D ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j using prems(3)[where
Γ=Γ1⊕E⊃D⊕F⊃D] by (auto simp add:union-ac)
    then obtain j where b1: j≤n
                 and b2: Γ1 ⊕ E⊃D ⊕ F⊃D ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j by blast
    from b2 have Γ1 ⊕ (E∨∗F)⊃D ⊕ S⊃B⊕(T⊃B) ⇒ C ↓ j+1
       using provable.ImpLD[where Γ=Γ1⊕S⊃B⊕(T⊃B) and A=E and B=F and C=D and
D=C and n=j]
       by (auto simp add:union-ac)
    then have Γ''⊕S⊃B⊕(T⊃B) ⇒ C ↓ j+1 using eq2 by simp
     then have ∃ j≤n+1. Γ''⊕S⊃B⊕(T⊃B) ⇒ C ↓ j using b1 by (rule-tac x=j+1 in exI)
(auto)
   }
   ultimately
   show ?case by blast
qed
```

lemma *inversionImpLL*:
assumes $\Gamma \oplus (S \supset T) \supset B \Rightarrow C \downarrow n$
shows $\exists\ j \leq n.\ \Gamma \oplus B \Rightarrow C \downarrow j$
using *assms*
proof (*induct* $\Gamma \equiv \Gamma \oplus (S \supset T) \supset B\ C\ n$ *arbitrary*:Γ)
case ($Ax\ i'\ \Gamma'$)
then have $At\ i' :\#\ \Gamma$ **by** *auto*
then have $\Gamma \oplus B \Rightarrow At\ i' \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case ($LBot\ \Gamma'\ C$)
then have $ff :\#\ \Gamma$ **by** *auto*
then have $\Gamma \oplus B \Rightarrow C \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case ($ImpR\ \Gamma'\ E\ F\ k$)
then have $\Gamma' \oplus E = \Gamma \oplus (S \supset T) \supset B \oplus E$ **by** *auto*
then have $\exists\ j.\ j \leq k \wedge \Gamma \oplus B \oplus E \Rightarrow F \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma \oplus E$] **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *c1*: $j \leq k$
and *c2*: $\Gamma \oplus B \oplus E \Rightarrow F \downarrow j$ **by** *auto*
from *c2* **have** $\Gamma \oplus B \Rightarrow E \supset F \downarrow j+1$ **using** *provable.ImpR*[**where** $\Gamma = \Gamma \oplus B$ **and** $A = E$ **and** $B = F$] **by** *auto*
then show *?case* **using** *c1* **by** *auto*
next
case ($ConjR\ \Gamma'\ E\ k\ F\ l$)
then have $\exists j \leq k.\ \Gamma \oplus B \Rightarrow E \downarrow j$ **and** $\exists j \leq l.\ \Gamma \oplus B \Rightarrow F \downarrow j$ **by** *auto*
then obtain *j1 j2* **where** *c1*: $j1 \leq k$
and *c2*: $\Gamma \oplus B \Rightarrow E \downarrow j1$
and *c3*: $j2 \leq l$
and *c4*: $\Gamma \oplus B \Rightarrow F \downarrow j2$ **by** *auto*
then show *?case* **using** *provable.ConjR*[**where** $\Gamma = \Gamma \oplus B$ **and** $n = j1$ **and** $m = j2$ **and** $A = E$ **and** $B = F$]
apply (*rule-tac* $x = j1+j2+1$ **in** *exI*) **by** *auto*
next
case ($ConjL\ \Gamma'\ E\ F\ C\ n\ \Gamma''$)
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \supset T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \wedge * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $A = E \wedge * F$ **and** $\Gamma' = \Gamma''$ **and** $B = (S \supset T) \supset B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus E \oplus F$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
then obtain *j* **where** *eq3*: $j \leq n$ **and** $\Gamma 1 \oplus E \oplus F \oplus B \Rightarrow C \downarrow j$ **by** *blast*

then have $\Gamma 1 \oplus E \wedge * F \oplus B \Rightarrow C \downarrow j+1$ **using** *provable.ConjL*[**where** $\Gamma = \Gamma 1 \oplus B$ **and** $A = E$ **and** $B = F$] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2 eq3* **by** *auto*
next
case (*DisjR1* Γ' E n F)
then have $\exists j \leq n.\ \Gamma \oplus B \Rightarrow E \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j \leq n$ **and** $\Gamma \oplus B \Rightarrow E \downarrow j$ **by** *blast*
then have $\Gamma \oplus B \Rightarrow E \vee * F \downarrow j+1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjR2* Γ' F n E)
then have $\exists j \leq n.\ \Gamma \oplus B \Rightarrow F \downarrow j$ **by** *auto*
then obtain j **where** *eq*:$j \leq n$ **and** $\Gamma \oplus B \Rightarrow F \downarrow j$ **by** *blast*
then have $\Gamma \oplus B \Rightarrow E \vee * F \downarrow j+1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*DisjL* Γ' E C n F m Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \supset T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus E \vee * F$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = E \vee * F$ **and** $B = (S \supset T) \supset B$] **by** *auto*
from *eq1 prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus E$] **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow j$ **by** (*auto simp add*:*union-ac*)
moreover
from *eq1 prems*(*5*)[**where** $\Gamma = \Gamma 1 \oplus F$] **have** $\exists\ k \leq m.\ \Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k$ **by** (*auto simp add*:*union-ac*)
ultimately
obtain j k **where** *a*: $j \leq n \wedge k \leq m$
and *b*: $\Gamma 1 \oplus E \oplus B \Rightarrow C \downarrow j$
and *c*: $\Gamma 1 \oplus F \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b c* **have** $\Gamma 1 \oplus E \vee * F \oplus B \Rightarrow C \downarrow j+k+1$ **using** *provable.DisjL*[**where** $\Gamma = \Gamma 1 \oplus B$ **and** $A = E$ **and** $B = F$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a eq2* **apply** (*rule-tac* $x = j+k+1$ **in** *exI*) **by** *auto*
next
case (*ImpLC* Γ' E F D C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S \supset T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E \wedge * F) \supset D$ **using** *midMultiset*[**where** $\Gamma = \Gamma'$ **and** $\Gamma' = \Gamma''$ **and** $A = (E \wedge * F) \supset D$ **and** $B = (S \supset T) \supset B$] **by** *auto*
from *eq1* **have** $\exists\ j \leq n.\ \Gamma 1 \oplus E \supset (F \supset D) \oplus B \Rightarrow C \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma = \Gamma 1 \oplus E \supset (F \supset D)$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j \leq n$
and *b2*: $\Gamma 1 \oplus E \supset (F \supset D) \oplus B \Rightarrow C \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (E \wedge * F) \supset D \oplus B \Rightarrow C \downarrow j+1$ **using** *provable.ImpLC*[**where** $\Gamma = \Gamma 1 \oplus B$ **and**

$A{=}E$ **and** $B{=}F$ **and** $C{=}D$ **and** $D{=}C$ **and** $n{=}j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma''{\oplus}B \Rightarrow C \downarrow j{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x{=}j{+}1$ **in** *exI*) (*auto*)
next
case (*ImpLD* Γ' E D F C n Γ'')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma' = \Gamma 1 \oplus (S{\supset}T) \supset B$
and *eq2*: $\Gamma'' = \Gamma 1 \oplus (E{\vee}{*}F){\supset}D$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}(E{\vee}{*}F){\supset}D$ **and** $B{=}(S{\supset}T) \supset B$] **by** *auto*
from *eq1* **have** $\exists\, j{\leq}n.\ \Gamma 1 \oplus E{\supset}D \oplus F{\supset}D \oplus B \Rightarrow C \downarrow j$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}E{\supset}D{\oplus}F{\supset}D$] **by** (*auto simp add*:*union-ac*)
then obtain j **where** *b1*: $j{\leq}n$
and *b2*: $\Gamma 1 \oplus E{\supset}D \oplus F{\supset}D \oplus B \Rightarrow C \downarrow j$ **by** *blast*
from *b2* **have** $\Gamma 1 \oplus (E{\vee}{*}F){\supset}D \oplus B \Rightarrow C \downarrow j{+}1$ **using** *provable.ImpLD*[**where** $\Gamma{=}\Gamma 1{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}D$ **and** $D{=}C$ **and** $n{=}j$]
by (*auto simp add*:*union-ac*)
then have $\Gamma''{\oplus}B \Rightarrow C \downarrow j{+}1$ **using** *eq2* **by** *simp*
then show *?case* **using** *b1* **by** (*rule-tac* $x{=}j{+}1$ **in** *exI*) (*auto*)
next
case (*ImpL0* Γ' j D C n Γ'')
then obtain $\Gamma 1$ **where** *eq1'*: $\Gamma' \oplus At\ j = \Gamma 1 \oplus (S{\supset}T) \supset B$
and *eq2'*: $\Gamma'' = \Gamma 1 \oplus At\ j \supset D$
using *midMultiset*[**where** $\Gamma{=}\Gamma' \oplus At\ j$ **and** $\Gamma'{=}\Gamma''$ **and** $A{=}At\ j \supset D$ **and** $B{=}(S{\supset}T) \supset B$]
by *auto*
from *eq1'* **obtain** $\Gamma 2$ **where** *eq1*: $\Gamma' = \Gamma 2 \oplus (S{\supset}T){\supset} B$
and *eq2''*: $\Gamma 1 = \Gamma 2 \oplus At\ j$ **using** *midMultiset*[**where** $\Gamma{=}\Gamma'$ **and** $\Gamma'{=}\Gamma 1$ **and** $A{=}At\ j$ **and** $B{=}(S{\supset}T){\supset} B$]
by *auto*
from *eq2''* *eq2'* **have** *eq2*: $\Gamma'' = \Gamma 2 \oplus At\ j \oplus At\ j \supset D$ **by** *simp*
from *eq1* **have** $\exists\ k{\leq}n.\ \Gamma 2 \oplus At\ j \oplus D \oplus B \Rightarrow C \downarrow k$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 2{\oplus}At\ j \oplus D$] **by** (*auto simp add*:*union-ac*)
then obtain k **where** *b1*: $k{\leq}n$
and *b2*: $\Gamma 2 \oplus At\ j \oplus D \oplus B \Rightarrow C \downarrow k$ **by** *blast*
from *b2* **have** $\Gamma 2 \oplus At\ j \oplus At\ j \supset D \oplus B \Rightarrow C \downarrow k{+}1$ **using** *provable.ImpL0*[**where** $\Gamma{=}\Gamma 2{\oplus}B$ **and** $i{=}j$ **and** $B{=}D$ **and** $C{=}C$ **and** $n{=}k$]
by (*auto simp add*:*union-ac*)
then have $\Gamma'' \oplus B \Rightarrow C \downarrow k{+}1$ **using** *eq2* **by** *simp*
then show $\exists\ j{\leq}n{+}1.\ \Gamma''{\oplus}B \Rightarrow C \downarrow j$ **using** *b1* **by** (*rule-tac* $x{=}k{+}1$ **in** *exI*) (*auto*)
next
case (*ImpLL* Γ' E F D n C m Γ'')
have $(E{\supset}F){\supset}D = (S{\supset}T){\supset}B \vee (E{\supset}F){\supset}D \neq (S{\supset}T){\supset}B$ **by** *blast*
moreover
{**assume** $(E{\supset}F){\supset}D = (S{\supset}T){\supset}B$

```
    then have E=S ∧ F=T ∧ D=B ∧ Γ'=Γ'' using prems by simp
    then have Γ'' ⊕ B ⇒ C ↓ m using prems by simp
    then have ∃ j≤n+m+1. Γ''⊕B ⇒ C ↓ j by (rule-tac x=m in exI) (auto)
   }
   moreover
   {assume (E⊃F)⊃D ≠ (S⊃T)⊃B
    then obtain Γ1 where eq1: Γ' = Γ1 ⊕ (S⊃T) ⊃ B
                     and eq2: Γ'' = Γ1 ⊕ (E⊃F)⊃D
         using midMultiset[where Γ=Γ' and Γ'=Γ'' and A=(E⊃F)⊃D and B=(S⊃T) ⊃ B]
prems by auto
   from eq1 have ∃ j≤n. Γ1 ⊕ E ⊕ F⊃D ⊕ B ⇒ F ↓ j using prems(3)[where Γ=Γ1⊕E⊕F⊃D]
by (auto simp add:union-ac)
    then obtain j where b1: j≤n
                    and b2: Γ1 ⊕ E ⊕ F⊃D ⊕ B ⇒ F ↓ j by blast
    moreover
    from eq1 have ∃ k≤m. Γ1 ⊕ D ⊕ B ⇒ C ↓ k using prems(5)[where Γ=Γ1⊕D] by (auto
simp add:union-ac)
    then obtain k where c1: k≤m
                    and c2: Γ1 ⊕ D ⊕ B ⇒ C ↓ k by blast
    ultimately
     have Γ1 ⊕ (E⊃F)⊃D ⊕ B ⇒ C ↓ j+k+1 using provable.ImpLL[where Γ=Γ1⊕B and
A=E and B=F and C=D and D=C and n=j and m=k]
       by (auto simp add:union-ac)
    then have Γ'' ⊕ B ⇒ C ↓ j+k+1 using eq2 by simp
     then have ∃ j≤n+m+1. Γ''⊕ B ⇒ C ↓ j using b1 c1 by (rule-tac x=j+k+1 in exI)
(auto)
   }
   ultimately
   show ?case by blast
qed

lemma inversionImpR:
  assumes Γ ⇒ A⊃B ↓ n
  shows ∃ j≤n. Γ ⊕ A ⇒ B ↓ j
  using assms
proof (induct Γ C≡A⊃B n)
  case (Ax i Γ)
  then show ?case by auto
next
  case (LBot Γ C)
  then have Γ ⊕ A ⇒ B ↓ 0 by auto
  then show ?case by blast
next
```

case (*ConjR* Γ *E n F m*)
then show *?case* **by** *auto*
next
case (*DisjR1* Γ *E n F*)
then show *?case* **by** *auto*
next
case (*DisjR2* Γ *F n E*)
then show *?case* **by** *auto*
next
case (*ConjL* Γ *E F C n*)
then have ∃*j*≤*n*. Γ ⊕ *E* ⊕ *F* ⊕ *A* ⇒ *B* ↓ *j* **by** *auto*
then obtain *j* **where** *a*: *j*≤*n*
and *b*: Γ ⊕ *E* ⊕ *F* ⊕ *A* ⇒ *B* ↓ *j* **by** *blast*
from *b* **have** Γ ⊕ *E*∧∗*F* ⊕ *A* ⇒ *B* ↓ *j*+*1* **using** *provable.ConjL*[**where** Γ=Γ⊕*A* **and** *A*=*E* **and** *B*=*F* **and** *C*=*B* **and** *n*=*j*] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*j*+*1* **in** *exI*) (*auto*)
next
case (*DisjL* Γ *E C n F m*)
then have ∃*j*≤*n*. Γ ⊕ *E* ⊕ *A* ⇒ *B* ↓ *j* **and** ∃*k*≤*m*. Γ ⊕ *F* ⊕ *A* ⇒ *B* ↓ *k* **by** *auto*
then obtain *j k* **where** *a*: *j*≤*n* ∧ *k*≤*m*
and *b*: Γ ⊕ *E* ⊕ *A* ⇒ *B* ↓ *j*
and *c*: Γ ⊕ *F* ⊕ *A* ⇒ *B* ↓ *k* **by** *blast*
from *b c* **have** Γ ⊕ *E*∨∗*F* ⊕ *A* ⇒ *B* ↓ *j*+*k*+*1* **using** *provable.DisjL*[**where** Γ=Γ⊕*A* **and** *A*=*E* **and** *B*=*F* **and** *C*=*B* **and** *n*=*j* **and** *m*=*k*]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*j*+*k*+*1* **in** *exI*) (*auto*)
next
case (*ImpL0* Γ *i D C n*)
then have ∃ *j*≤*n*. Γ ⊕ *At i* ⊕ *D* ⊕ *A* ⇒ *B* ↓ *j* **by** *auto*
then obtain *j* **where** *a*: *j*≤*n*
and *b*: Γ ⊕ *At i* ⊕ *D* ⊕ *A* ⇒ *B* ↓ *j* **by** *blast*
from *b* **have** Γ ⊕ *At i* ⊕ *At i* ⊃ *D* ⊕ *A* ⇒ *B* ↓ *j*+*1* **using** *provable.ImpL0*[**where** Γ=Γ⊕*A* **and** *i*=*i* **and** *B*=*D* **and** *C*=*B* **and** *n*=*j*]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*j*+*1* **in** *exI*) (*auto*)
next
case (*ImpLC* Γ *E F D C n*)
then obtain *j* **where** *a*: *j*≤*n*
and *b*: Γ ⊕ *E*⊃(*F*⊃*D*) ⊕ *A* ⇒ *B* ↓ *j* **by** *auto*
from *b* **have** Γ ⊕ (*E*∧∗*F*)⊃*D* ⊕ *A* ⇒ *B* ↓ *j*+*1* **using** *provable.ImpLC*[**where** Γ=Γ⊕*A* **and** *A*=*E* **and** *B*=*F* **and** *C*=*D* **and** *D*=*B* **and** *n*=*j*]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*j*+*1* **in** *exI*) (*auto*)

next
case (*ImpLD* Γ *E D F C n*)
then obtain *j* **where** *a*: $j \leq n$
and *b*: $\Gamma \oplus E \supset D \oplus F \supset D \oplus A \Rightarrow B \downarrow j$ **by** *blast*
from *b* **have** $\Gamma \oplus (E \vee * F) \supset D \oplus A \Rightarrow B \downarrow j+1$ **using** *provable.ImpLD*[**where** $\Gamma = \Gamma \oplus A$ **and** $A = E$ **and** $B = F$ **and** $C = D$ **and** $D = B$ **and** $n = j$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*j*+*1* **in** *exI*) (*auto*)
next
case (*ImpLL* Γ *E F D n C m*)
then obtain *k* **where** *a*: $k \leq m$
and *b*: $\Gamma \oplus D \oplus A \Rightarrow B \downarrow k$ **by** *auto*
from *prems* **have** $\Gamma \oplus E \oplus F \supset D \oplus A \Rightarrow F \downarrow n$ **using** *dpWeak* **by** *auto*
then have $\Gamma \oplus (E \supset F) \supset D \oplus A \Rightarrow B \downarrow n+k+1$ **using** *provable.ImpLL*[**where** $\Gamma = \Gamma \oplus A$ **and** $A = E$ **and** $B = F$ **and** $C = D$ **and** $n = n$ **and** $D = B$ **and** $m = k$] *b*
by (*auto simp add*:*union-ac*)
then show *?case* **using** *a* **by** (*rule-tac x*=*n*+*k*+*1* **in** *exI*) (*auto*)
next
case (*ImpR* Γ *E F n*)
have $E \supset F = A \supset B \vee E \supset F \neq A \supset B$ **by** *blast*
moreover
{assume $E \supset F = A \supset B$
then have $\Gamma \oplus A \Rightarrow B \downarrow n$ **using** *prems* **by** *auto*
then have $\exists\ j \leq n+1.\ \Gamma \oplus A \Rightarrow B \downarrow j$ **by** (*rule-tac x*=*n* **in** *exI*) (*auto*)
}
moreover
{assume $E \supset F \neq A \supset B$
then have $\exists\ j \leq n+1.\ \Gamma \oplus A \Rightarrow B \downarrow j$ **using** *prems* **by** *auto*
}
ultimately
show *?case* **by** *blast*
qed

lemma *genAx*:
assumes $w = weight\ A$
shows $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$
using *assms*
proof (*induct w arbitrary*: *A rule*:*nat-less-induct*)
case *1*
then show *?case*
proof (*cases A*)
case (*At i*)
then have $\Gamma \oplus At\ i \Rightarrow At\ i \downarrow 0$ **by** *auto*

then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** *blast*
next
case *ff*
then have $\Gamma \oplus ff \Rightarrow ff \downarrow 0$ **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** *blast*
next
case (*Conj E F*)
then have *a*: *weight E* < *weight A* ∧ *weight F* < *weight A* **by** *auto*
from *a* **have** $\exists\ n.\ \Gamma \oplus E \Rightarrow E \downarrow n$ **using** *prems*(*2*,*3*) **by** *auto*
moreover
from *a* **have** $\exists\ m.\ \Gamma \oplus F \Rightarrow F \downarrow m$ **using** *prems*(*2*,*3*) **by** *auto*
ultimately
obtain *n m* **where** *b1*: $\Gamma \oplus E \Rightarrow E \downarrow n$
and *b2*: $\Gamma \oplus F \Rightarrow F \downarrow m$ **by** *blast*
from *b1* **have** $\Gamma \oplus E \oplus F \Rightarrow E \downarrow n$ **using** *dpWeak* **by** *auto*
moreover
from *b2* **have** $\Gamma \oplus E \oplus F \Rightarrow F \downarrow m$ **using** *dpWeak*[**where** Γ=$\Gamma\oplus F$ **and** *A*=*E* **and** *C*=*F* **and** *n*=*m*] **by** (*auto simp add*:*union-ac*)
ultimately
have $\Gamma \oplus E \oplus F \Rightarrow E\wedge *F \downarrow n+m+1$ **by** (*rule provable.ConjR*)
then have $\Gamma \oplus E\wedge *F \Rightarrow E\wedge *F \downarrow n+m+2$ **using** *provable.ConjL*[**where** $C=E\wedge *F$ **and** *A*=*E* **and** *B*=*F* **and** *n*=*n*+*m*+*1*] **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** (*rule-tac x*=*n*+*m*+*2* **in** *exI*) (*auto*)
next
case (*Disj E F*)
then have *a*: *weight E* < *weight A* ∧ *weight F* < *weight A* **by** *auto*
from *a* **have** $\exists\ n.\ \Gamma \oplus E \Rightarrow E \downarrow n$ **using** *prems*(*2*,*3*) **by** *auto*
moreover
from *a* **have** $\exists\ m.\ \Gamma \oplus F \Rightarrow F \downarrow m$ **using** *prems*(*2*,*3*) **by** *auto*
ultimately
obtain *n m* **where** *b1*: $\Gamma \oplus E \Rightarrow E \downarrow n$
and *b2*: $\Gamma \oplus F \Rightarrow F \downarrow m$ **by** *blast*
from *b1* **have** $\Gamma \oplus E \Rightarrow E\vee *F \downarrow n+1$ **using** *provable.DisjR1* **by** *auto*
moreover
from *b2* **have** $\Gamma \oplus F \Rightarrow E\vee *F \downarrow m+1$ **using** *provable.DisjR2* **by** *auto*
ultimately
have $\Gamma \oplus E\vee *F \Rightarrow E\vee *F \downarrow n+1+\ (m+1)\ +1$ **using** *provable.DisjL*[**where** $C=E\vee *F$ **and** *n*=*n*+*1* **and** *m*=*m*+*1*] **by** *arith*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** (*rule-tac x*=*n*+*1*+(*m*+*1*)+*1* **in** *exI*) (*auto*)
next
case (*Imp E F*)
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** ‹A=$E\supset F$›

proof (*cases E*)
case (*At i*)
have *weight F* < *weight A* **using** ‹*A*=*E*⊃*F*› **by** *auto*
then have ∃ *n*. Γ ⊕ *F* ⇒ *F* ↓ *n* **using** *prems*(*2*,*3*) **by** *auto*
then obtain *n* **where** *a*: Γ ⊕ *F* ⇒ *F* ↓ *n* **by** *blast*
from *a* **have** Γ ⊕ *At i* ⊕ *F* ⇒ *F* ↓ *n* **using** *dpWeak*[**where** Γ=Γ⊕*F* **and** *A*=*At i*] **by** (*auto simp add*:*union-ac*)
then have Γ ⊕ *At i* ⊕ *At i* ⊃ *F* ⇒ *F* ↓ *n*+*1* **using** *provable.ImpL0* **by** *auto*
then have Γ ⊕ *At i* ⊃ *F* ⇒ *At i* ⊃ *F* ↓ *n*+*2* **using** *provable.ImpR*[**where** Γ=Γ⊕ *At i* ⊃ *F*] **by** (*auto simp add*:*union-ac*)
then show ∃ *n*. Γ ⊕ *A* ⇒ *A* ↓ *n* **using** *prems* **by** *auto*
next
case *ff*
have Γ ⊕ *ff* ⊃ *F* ⊕ *ff* ⇒ *F* ↓ *0* **by** *auto*
then have Γ ⊕ *ff* ⊃ *F* ⇒ *ff* ⊃ *F* ↓ *1* **using** *provable.ImpR*[**where** Γ=Γ ⊕ *ff* ⊃ *F*] **by** *auto*
then show ∃ *n*. Γ ⊕ *A* ⇒ *A* ↓ *n* **using** *prems* **by** *auto*
next
case (*Imp G H*)
then have *weight* (*G*⊃*H*) < *weight A* **using** ‹*A*=*E*⊃*F*› **by** *auto*
then have ∃ *n*. Γ ⊕ *G*⊃*H* ⇒ *G*⊃*H* ↓ *n* **using** *prems*(*2*,*3*) **by** *auto*
then obtain *n* **where** *a*: Γ ⊕ *G*⊃*H* ⇒ *G*⊃*H* ↓ *n* **by** *blast*
from *a* **have** Γ ⊕ *G*⊃*H* ⊕ *H*⊃*F* ⇒ *G*⊃*H* ↓ *n* **using** *dpWeak* **by** *auto*
then have ∃ *n*′≤*n*. Γ ⊕ *G*⊃*H* ⊕ *H*⊃*F* ⊕ *G* ⇒ *H* ↓ *n*′ **using** *inversionImpR*[**where** Γ=Γ ⊕ *G*⊃*H* ⊕ *H*⊃*F* **and** *A*=*G* **and** *B*=*H*] **by** *auto*
then obtain *n*′ **where** Γ ⊕ *G*⊃*H* ⊕ *H*⊃*F* ⊕ *G* ⇒ *H* ↓ *n*′ **by** *blast*
moreover
have *weight F* < *weight A* **using** ‹*A*=*E*⊃*F*› **by** *auto*
then have ∃ *m*. Γ ⊕ *F* ⇒ *F* ↓ *m* **using** *prems*(*2*,*3*) **by** *auto*
then obtain *m* **where** Γ ⊕ *F* ⇒ *F* ↓ *m* **by** *blast*
then have Γ ⊕ *F* ⊕ *G*⊃*H* ⇒ *F* ↓ *m* **using** *dpWeak* **by** *auto*
ultimately
have Γ ⊕ (*G*⊃*H*)⊃*F* ⊕ *G*⊃*H* ⇒ *F* ↓ *n*′+*m*+*1* **using** *provable.ImpLL*[**where** Γ=Γ⊕*G*⊃*H* **and** *C*=*F* **and** *A*=*G* **and** *B*=*H* **and** *n*=*n*′ **and** *m*=*m*]
by (*auto simp add*:*union-ac*)
then have Γ ⊕ (*G*⊃*H*)⊃*F* ⇒ (*G*⊃*H*)⊃*F* ↓ *n*′+*m*+*2* **using** *provable.ImpR* **by** *auto*
then show ∃ *n*. Γ ⊕ *A* ⇒ *A* ↓ *n* **using** *prems* **by** *auto*
next
case (*Conj G H*)
then have *weight* (*G*⊃(*H*⊃*F*)) < *weight A* **using** ‹*A*=*E*⊃*F*› **by** *auto*
then have ∃ *n*. Γ ⊕ *G*⊃(*H*⊃*F*) ⇒ *G*⊃(*H*⊃*F*) ↓ *n* **using** *prems*(*2*,*3*) **by** *auto*
then have ∃ *n*. Γ ⊕ *G*⊃(*H*⊃*F*) ⊕ *G* ⇒ *H*⊃*F* ↓ *n* **using** *inversionImpR*[**where**

$\Gamma{=}\Gamma{\oplus}G{\supset}(H{\supset}F)$ **and** $A{=}G$ **and** $B{=}H{\supset}F]$ **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G{\supset}(H{\supset}F) \oplus G \oplus H \Rightarrow F \downarrow n$ **using** *inversionImpR*[**where** $\Gamma{=}\Gamma \oplus G{\supset}(H{\supset}F) \oplus G$ **and** $A{=}H$ **and** $B{=}F$] **by** *auto*
then obtain n **where** a: $\Gamma \oplus G{\supset}(H{\supset}F) \oplus G \oplus H \Rightarrow F \downarrow n$ **by** *blast*
from a **have** $\Gamma \oplus (G{\wedge}{*}H){\supset}F \oplus G \oplus H \Rightarrow F \downarrow n{+}1$ **using** *provable.ImpLC*[**where** $\Gamma{=}\Gamma \oplus G \oplus H$ **and** $A{=}G$ **and** $B{=}H$ **and** $C{=}F$ **and** $D{=}F$]
by (*auto simp add*:*union-ac*)
then have $\Gamma \oplus (G{\wedge}{*}H){\supset}F \oplus G{\wedge}{*}H \Rightarrow F \downarrow n{+}2$
using *provable.ConjL*[**where** $\Gamma{=}\Gamma \oplus (G{\wedge}{*}H){\supset}F$ **and** $A{=}G$ **and** $B{=}H$ **and** $C{=}F$ **and** $n{=}n{+}1$] **by** (*auto simp add*:*union-ac*)
then have $\Gamma \oplus (G{\wedge}{*}H){\supset}F \Rightarrow (G{\wedge}{*}H){\supset}F \downarrow n{+}2{+}1$
using *provable.ImpR*[**where** $\Gamma{=}\Gamma \oplus (G{\wedge}{*}H){\supset}F$ **and** $A{=}G{\wedge}{*}H$ **and** $B{=}F$ **and** $n{=}n{+}2$] **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** *auto*
next
case (*Disj G H*)
then have *weight* $(G{\supset}F) <$ *weight* A **using** ‹$A{=}E{\supset}F$› **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G{\supset}F \Rightarrow G{\supset}F \downarrow n$ **using** *prems*(*2*,*3*) **by** *auto*
then have $\exists\ n.\ \Gamma \oplus G{\supset}F \oplus H{\supset}F \Rightarrow G{\supset}F \downarrow n$ **using** *dpWeak* **by** *blast*
then have $\exists\ n.\ \Gamma \oplus G{\supset}F \oplus H{\supset}F \oplus G \Rightarrow F \downarrow n$ **using** *inversionImpR*[**where** $\Gamma{=}\ \Gamma{\oplus}G{\supset}F{\oplus}H{\supset}F$ **and** $A{=}G$ **and** $B{=}F$] **by** *auto*
then obtain n **where** $\Gamma \oplus G{\supset}F \oplus H{\supset}F \oplus G \Rightarrow F \downarrow n$ **by** *blast*
moreover
have *weight* $(H{\supset}F) <$ *weight* A **using** ‹$A{=}E{\supset}F$› **and** ‹$E{=}G{\vee}{*}H$› **by** *auto*
then have $\exists\ n.\ \Gamma \oplus H{\supset}F \Rightarrow H{\supset}F \downarrow n$ **using** *prems*(*2*,*3*) **by** *auto*
then have $\exists\ n.\ \Gamma \oplus H{\supset}F \oplus G{\supset}F \Rightarrow H{\supset}F \downarrow n$ **using** *dpWeak* **by** *blast*
then have $\exists\ n.\ \Gamma \oplus G{\supset}F \oplus H{\supset}F \oplus H \Rightarrow F \downarrow n$ **using** *inversionImpR*[**where** $\Gamma{=}\ \Gamma{\oplus}G{\supset}F{\oplus}H{\supset}F$ **and** $A{=}H$ **and** $B{=}F$]
by (*auto simp add*:*union-ac*)
then obtain m **where** $\Gamma \oplus G{\supset}F \oplus H{\supset}F \oplus H \Rightarrow F \downarrow m$ **by** *blast*
ultimately
have $\Gamma \oplus G{\supset}F \oplus H{\supset}F \oplus G{\vee}{*}H \Rightarrow F \downarrow n{+}m{+}1$ **using** *provable.DisjL*[**where** $\Gamma{=}\Gamma{\oplus}G{\supset}F{\oplus}H{\supset}F$ **and** $A{=}G$ **and** $B{=}H$] **by** *auto*
then have $\Gamma \oplus (G{\vee}{*}H){\supset}F \oplus G{\vee}{*}H \Rightarrow F \downarrow n{+}m{+}2$ **using** *provable.ImpLD*[**where** $\Gamma{=}\Gamma{\oplus}G{\vee}{*}H$ **and** $A{=}G$ **and** $B{=}H$ **and** $C{=}F$]
by (*auto simp add*:*union-ac*)
then have $\Gamma \oplus (G{\vee}{*}H){\supset}F \Rightarrow (G{\vee}{*}H){\supset}F \downarrow n{+}m{+}2{+}1$ **using** *provable.ImpR*[**where** $\Gamma{=}\Gamma{\oplus}(G{\vee}{*}H){\supset}F$ **and** $A{=}G{\vee}{*}H$ **and** $B{=}F$ **and** $n{=}n{+}m{+}2$]
by *arith*
then show $\exists\ n.\ \Gamma \oplus A \Rightarrow A \downarrow n$ **using** *prems* **by** *blast*
qed
qed
qed

lemma *modusPonens*:
shows $\exists\ n.\ \Gamma \oplus A \oplus A{\supset}B \Rightarrow B \downarrow n$
proof−
have $\exists n.\ \Gamma \oplus A{\supset}B \Rightarrow A{\supset}B \downarrow n$ **using** *genAx*[**where** $A{=}A{\supset}B$] **by** *auto*
then show $\exists\ n.\ \Gamma \oplus A \oplus A{\supset}B \Rightarrow B \downarrow n$ **using** *inversionImpR*[**where** $\Gamma{=}\Gamma{\oplus}A{\supset}B$ **and** $A{=}A$ **and** $B{=}B$] **by** (*auto simp add*:*union-ac*)
qed

lemma *ImpLClassical*:
fixes B :: *form*
assumes $\Gamma \Rightarrow D \downarrow n$ **and** $\exists\ m.\ \Gamma \oplus B \Rightarrow S \downarrow m$
shows $\exists\ n.\ \Gamma \oplus D{\supset}B \Rightarrow S \downarrow n$
using *assms*
proof (*induct arbitrary*:*B*)
case (*Ax i* Γ)
then have $\exists\ \Gamma'.\ \Gamma = \Gamma' \oplus At\ i$ **using** *containMultiset* **by** *auto*
then obtain Γ' **where** *eq*: $\Gamma = \Gamma' \oplus At\ i$ **by** *blast*
then have $\exists\ m.\ \Gamma' \oplus At\ i \oplus B \Rightarrow S \downarrow m$ **using** *prems* **by** *simp*
then have $\exists\ m.\ \Gamma' \oplus At\ i \oplus At\ i \supset B \Rightarrow S \downarrow m$ **using** *provable.ImpL0* **by** *auto*
then show *?case* **using** *eq* **by** *simp*
next
case (*LBot* Γ *C*)
then have $\Gamma \oplus C{\supset}B \Rightarrow S \downarrow 0$ **by** *auto*
then show *?case* **by** *blast*
next
case (*ConjL* Γ *E F C n*)
then obtain *a* **where** $\Gamma \oplus E \wedge{*}\ F \oplus B \Rightarrow S \downarrow a$ **by** *blast*
then have $\exists\ m.\ \Gamma \oplus E \oplus F \oplus B \Rightarrow S \downarrow m$ **using** *inversionConjL*[**where** $\Gamma{=}\Gamma{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}S$ **and** $B{=}B$ **and** $n{=}a$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ m.\ \Gamma \oplus E \oplus F \oplus C{\supset}B \Rightarrow S \downarrow m$ **using** *prems*(*4*) **by** *simp*
then show *?case* **using** *provable.ConjL*[**where** $\Gamma{=}\Gamma{\oplus}C{\supset}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}S$] **by** (*auto simp add*:*union-ac*)
next
case (*DisjL* Γ *E C n F m*)
then obtain *a* **where** $\Gamma \oplus E \vee{*}\ F \oplus B \Rightarrow S \downarrow a$ **by** *blast*
then have $\exists\ n.\ \Gamma \oplus E \oplus B \Rightarrow S \downarrow n$ **and** $\exists\ m.\ \Gamma \oplus F \oplus B \Rightarrow S \downarrow m$ **using** *inversionDisjL*[**where** $\Gamma{=}\Gamma{\oplus}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}S$ **and** $n{=}a$]
by (*auto simp add*:*union-ac*)
then have $\exists\ n.\ \Gamma{\oplus}E{\oplus}\ C{\supset}B \Rightarrow S \downarrow n$ **and** $\exists\ m.\ \Gamma{\oplus}F \oplus C{\supset}B \Rightarrow S \downarrow m$ **using** *prems*(*4*,*6*) **by** *auto*
then show *?case* **using** *provable.DisjL*[**where** $\Gamma{=}\Gamma{\oplus}C{\supset}B$ **and** $A{=}E$ **and** $B{=}F$ **and** $C{=}S$] **by** (*auto simp add*:*union-ac*)

next
case (*ImpL0* Γ *i E C n*)
then obtain *a* **where** Γ ⊕ *At i* ⊕ *At i*⊃*E* ⊕ *B* ⇒ *S* ↓ *a* **by** *blast*
then have ∃ *m*. Γ ⊕ *At i* ⊕ *E* ⊕ *B* ⇒ *S* ↓ *m* **using** *inversionImpL0*[**where** Γ=Γ⊕*At i*⊕*B* **and** *i*=*i* **and** *B*=*E* **and** *C*=*S* **and** *n*=*a*] **by** (*auto simp add*:*union-ac*)
then have ∃ *m*. Γ ⊕ *At i* ⊕ *E* ⊕ *C*⊃*B* ⇒ *S* ↓ *m* **using** *prems*(*4*) **by** *simp*
then show *?case* **using** *provable.ImpL0*[**where** Γ=Γ⊕*C*⊃*B* **and** *i*=*i* **and** *B*=*E* **and** *C*=*S*] **by** (*auto simp add*:*union-ac*)
next
case (*ImpLC* Γ *E F G C n*)
then obtain *a* **where** Γ ⊕ (*E*∧∗*F*)⊃*G* ⊕ *B* ⇒ *S* ↓ *a* **by** *blast*
then have ∃ *m*. Γ ⊕ *E*⊃(*F*⊃*G*)⊕ *B* ⇒ *S* ↓ *m* **using** *inversionImpLC*[**where** Γ=Γ⊕*B* **and** *S*=*E* **and** *T*=*F* **and** *B*=*G* **and** *C*=*S* **and** *n*=*a*] **by** (*auto simp add*:*union-ac*)
then have ∃ *m*. Γ ⊕ *E*⊃(*F*⊃*G*)⊕ *C*⊃*B* ⇒ *S* ↓ *m* **using** *prems*(*4*) **by** *simp*
then show *?case* **using** *provable.ImpLC*[**where** Γ=Γ⊕*C*⊃*B* **and** *A*=*E* **and** *B*=*F* **and** *C*=*G* **and** *D*=*S*] **by** (*auto simp add*:*union-ac*)
next
case (*ImpLD* Γ *E G F C n*)
then obtain *a* **where** Γ ⊕ (*E*∨∗*F*)⊃*G* ⊕ *B* ⇒ *S* ↓ *a* **by** *blast*
then have ∃ *m*. Γ ⊕ *E*⊃*G* ⊕ *F*⊃*G* ⊕ *B* ⇒ *S* ↓ *m* **using** *inversionImpLD*[**where** Γ=Γ⊕*B* **and** *S*=*E* **and** *T*=*F* **and** *B*=*G* **and** *C*=*S* **and** *n*=*a*]
by (*auto simp add*:*union-ac*)
then have ∃ *m*. Γ ⊕ *E*⊃*G* ⊕ *F*⊃*G* ⊕ *C*⊃*B* ⇒ *S* ↓ *m* **using** *prems*(*4*) **by** *simp*
then show *?case* **using** *provable.ImpLD*[**where** Γ=Γ⊕*C*⊃*B* **and** *A*=*E* **and** *C*=*G* **and** *B*=*F* **and** *D*=*S*] **by** (*auto simp add*:*union-ac*)
next
case (*ConjR* Γ *E n F m*)
then have ∃ *m*. Γ ⊕ *F*⊃*B* ⇒ *S* ↓ *m* **by** *auto*
then have ∃ *m*. Γ ⊕ *E*⊃(*F*⊃*B*) ⇒ *S* ↓ *m* **using** *prems*(*4*)[**where** *B*=*F*⊃*B*] **by** *auto*
then show *?case* **using** *provable.ImpLC* **by** *auto*
next
case (*DisjR1* Γ *E n F*)
then have ∃ *a*. Γ ⊕ *E*⊃*B* ⇒ *S* ↓ *a* **by** *auto*
then have ∃ *a*. Γ ⊕ *E*⊃*B* ⊕ *F*⊃*B* ⇒ *S* ↓ *a* **using** *dpWeak*[**where** Γ=Γ⊕*E*⊃*B* **and** *A*=*F*⊃*B*] **by** *auto*
then show *?case* **using** *provable.ImpLD*[**where** *A*=*E* **and** *C*=*B* **and** *B*=*F*] **by** *auto*
next
case (*DisjR2* Γ *F n E*)
then have ∃ *a*. Γ ⊕ *F*⊃*B* ⇒ *S* ↓ *a* **by** *auto*
then have ∃ *a*. Γ ⊕ *E*⊃*B* ⊕ *F*⊃*B* ⇒ *S* ↓ *a* **using** *dpWeak*[**where** Γ=Γ⊕*F*⊃*B* **and** *A*=*E*⊃*B*] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *provable.ImpLD*[**where** *A*=*E* **and** *C*=*B* **and** *B*=*F*] **by** *auto*
next

case (*ImpR* Γ *E F n*)
then have Γ ⊕ *E* ⊕ *F*⊃*B* ⇒ *F* ↓ *n* **using** *dpWeak*[**where** Γ=Γ⊕*E* **and** *A*=*F*⊃*B*] **by** *auto*
then show *?case* **using** *prems*(*5*) *provable.ImpLL* **by** *auto*
next
case (*ImpLL* Γ *F G H n E m*)
then obtain *a* **where** Γ ⊕ (*F*⊃*G*)⊃*H* ⊕ *B* ⇒ *S* ↓ *a* **by** *blast*
then have ∃ *m*. Γ ⊕ *H* ⊕ *B* ⇒ *S* ↓ *m* **using** *inversionImpLL*[**where** Γ=Γ⊕*B* **and** *S*=*F* **and** *T*=*G* **and** *B*=*H* **and** *C*=*S* **and** *n*=*a*] **by** (*auto simp add*:*union-ac*)
then have *a*: ∃ *a*. Γ ⊕ *H* ⊕ *E*⊃*B* ⇒ *S* ↓ *a* **using** *prems*(*6*) **by** *auto*
moreover
from *prems*(*3*) **have** Γ ⊕ *G*⊃*H* ⊕ *F* ⊕ *E*⊃*B* ⇒ *G* ↓ *n* **using** *dpWeak*[**where** Γ=Γ ⊕ *G*⊃*H* ⊕ *F* **and** *A*=*E*⊃*B*] **by** (*auto simp add*:*union-ac*)
ultimately
show *?case* **using** *provable.ImpLL*[**where** Γ=Γ⊕*E*⊃*B* **and** *A*=*F* **and** *B*=*G* **and** *C*=*H* **and** *n*=*n* **and** *D*=*S*] **by** (*auto simp add*:*union-ac*)
qed

lemma *twoDB*:
assumes Γ ⊕ (*C*⊃*D*)⊃*B* ⇒ *E* ↓ *n*
shows ∃ *m*. Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *E* ↓ *m*
using *assms*
proof (*induct* Γ≡Γ ⊕ (*C*⊃*D*)⊃*B E n arbitrary*: Γ)
case (*Ax i* Γ)
then have ∃ Γ*1*. Γ = Γ*1* ⊕ *At i* **using** *containMultiset* **by** *auto*
then obtain Γ*1* **where** *eq*: Γ = Γ*1* ⊕ *At i* **by** *blast*
then have Γ*1* ⊕ *At i* ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *At i* ↓ *0* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*LBot* Γ *E*)
then have ∃ Γ*1*. Γ = Γ*1* ⊕ *ff* **using** *containMultiset* **by** *auto*
then obtain Γ*1* **where** *eq*: Γ = Γ*1* ⊕ *ff* **by** *blast*
then have Γ*1* ⊕ *ff* ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *E* ↓ *0* **by** *auto*
then show *?case* **using** *eq* **by** *auto*
next
case (*ConjR* Γ *G n H m*)
then have ∃*n*. Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *G* ↓ *n* **and** ∃*m*. Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *H* ↓ *m* **by** *auto*
then obtain *n m* **where** Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *G* ↓ *n* **and** Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *H* ↓ *m* **by** *blast*
then have Γ ⊕ *C* ⊕ *D*⊃*B* ⊕ *D*⊃*B* ⇒ *G*∧∗*H* ↓ *n*+*m*+*1* **using** *provable.ConjR* **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR1* Γ *G n H*)

then obtain n **where** $\Gamma \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow G \downarrow n$ **by** *auto*
then have $\Gamma \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow G{\vee}{*}H \downarrow n{+}1$ **using** *provable.DisjR1* **by** *auto*
then show *?case* **by** *blast*
next
case (*DisjR2* Γ H n G)
then obtain n **where** $\Gamma \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow H \downarrow n$ **by** *auto*
then have $\Gamma \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow G{\vee}{*}H \downarrow n{+}1$ **using** *provable.DisjR2* **by** *auto*
then show *?case* **by** *blast*
next
case (*ConjL* Γ G H E n Γ')
then have $\exists\ \Gamma 1.\ \Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B \wedge \Gamma'{=}\Gamma 1 \oplus G{\wedge}{*}H$ **using** *midMultiset*[**where** $A{=}G{\wedge}{*}H$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus G{\wedge}{*}H$ **by** *auto*
from *eq1* **have** $\exists\ n.\ \Gamma 1 \oplus G \oplus H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1 \oplus G \oplus H$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ n.\ \Gamma 1 \oplus G{\wedge}{*}H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **using** *provable.ConjL*[**where** $\Gamma{=}\Gamma 1{\oplus}C{\oplus}D{\supset}B{\oplus}D{\supset}B$ **and** $A{=}G$ **and** $B{=}H$] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2* **by** *blast*
next
case (*DisjL* Γ G E n H m Γ')
then have $\exists\ \Gamma 1.\ \Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B \wedge \Gamma'{=}\Gamma 1 \oplus G{\vee}{*}H$ **using** *midMultiset*[**where** $A{=}G{\vee}{*}H$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus G{\vee}{*}H$ **by** *auto*
from *eq1* **have** $\exists\ n.\ \Gamma 1 \oplus G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1{\oplus}G$] **by** (*auto simp add*:*union-ac*)
moreover
from *eq1* **have** $\exists\ m.\ \Gamma 1 \oplus H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow m$ **using** *prems*(*5*)[**where** $\Gamma{=}\Gamma 1{\oplus}H$] **by** (*auto simp add*:*union-ac*)
ultimately
obtain n m **where** $\Gamma 1 \oplus G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **and** $\Gamma 1 \oplus H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow m$ **by** *auto*
then have $\Gamma 1 \oplus G{\vee}{*}H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n{+}m{+}1$ **using** *provable.DisjL*[**where** $\Gamma{=}\Gamma 1{\oplus}C{\oplus}D{\supset}B{\oplus}D{\supset}B$ **and** $A{=}G$ **and** $B{=}H$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2* **by** (*rule-tac* $x{=}n{+}m{+}1$ **in** *exI*) (*auto simp add*:*union-ac*)
next
case (*ImpR* Γ G H n Γ')
then have $\Gamma \oplus G = \Gamma' \oplus G \oplus (C{\supset}D){\supset}B$ **by** (*auto simp add*:*union-ac*)
then obtain n **where** $\Gamma' \oplus G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow H \downarrow n$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma'{\oplus}G$] **by** *auto*
then have $\Gamma' \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow G{\supset}H \downarrow n{+}1$ **using** *provable.ImpR*[**where** $\Gamma{=}\Gamma'{\oplus}C{\oplus}D{\supset}B{\oplus}D{\supset}B$

and $A{=}G$ **and** $B{=}H$] **by** (*auto simp add*:*union-ac*)
then show *?case* **by** *blast*
next
case (*ImpL0* Γ i G E n Γ')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma \oplus At\ i = \Gamma 1 \oplus (C{\supset}D){\supset}B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus (At\ i \supset G)$
using *midMultiset*[**where** $\Gamma{=}\Gamma \oplus At\ i$ **and** $A{=}At\ i \supset G$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
from *eq1* **obtain** $\Gamma 2$ **where** *eq3*: $\Gamma = \Gamma 2 \oplus (C{\supset}D){\supset}B$
and *eq4*: $\Gamma 1 = \Gamma 2 \oplus At\ i$
using *midMultiset*[**where** $A{=}At\ i$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
from *eq4 eq2* **have** *eq5*: $\Gamma' = \Gamma 2 \oplus At\ i \oplus (At\ i \supset G)$ **by** *simp*
from *eq3* **have** $\exists\ n.\ \Gamma 2 \oplus At\ i \oplus G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 2 \oplus At\ i \oplus G$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ n.\ \Gamma 2 \oplus At\ i \oplus At\ i \supset G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow n$ **using** *provable.ImpL0*[**where** $\Gamma{=}\Gamma 2 \oplus C \oplus D{\supset}B \oplus D{\supset}B$ **and** $B{=}G$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *eq5* **by** *blast*
next
case (*ImpLC* Γ F G H E n Γ')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus (F{\wedge}{*}G){\supset}H$
using *midMultiset*[**where** $\Gamma{=}\Gamma$ **and** $\Gamma'{=}\Gamma'$ **and** $A{=}(F{\wedge}{*}G){\supset}H$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
from *eq1* **have** $\exists\, a.\ \Gamma 1 \oplus F{\supset}(G{\supset}H) \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow a$ **using** *prems*(*3*)[**where** $\Gamma{=}\Gamma 1 \oplus F{\supset}(G{\supset}H)$] **by** (*auto simp add*:*union-ac*)
then have $\exists\, a.\ \Gamma 1 \oplus (F{\wedge}{*}G){\supset}H \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow a$ **using** *provable.ImpLC*[**where** $\Gamma{=}\Gamma 1 \oplus C \oplus D{\supset}B \oplus D{\supset}B$ **and** $A{=}F$ **and** $B{=}G$ **and** $C{=}H$]
by (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2* **by** *blast*
next
case (*ImpLD* Γ F G H E n Γ')
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus (C{\supset}D){\supset}B$
and *eq2*: $\Gamma' = \Gamma 1 \oplus (F{\vee}{*}H){\supset}G$
using *midMultiset*[**where** $\Gamma{=}\Gamma$ **and** $\Gamma'{=}\Gamma'$ **and** $A{=}(F{\vee}{*}H){\supset}G$ **and** $B{=}(C{\supset}D){\supset}B$] **by** *auto*
from *eq1* **have** $\exists\, a.\ \Gamma 1 \oplus F{\supset}G \oplus H{\supset}G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow a$
using *prems*(*3*)[**where** $\Gamma{=}\Gamma 1 \oplus F{\supset}G \oplus H{\supset}G$] **by** (*auto simp add*:*union-ac*)
then have $\exists\, a.\ \Gamma 1 \oplus (F{\vee}{*}H){\supset}G \oplus C \oplus D{\supset}B \oplus D{\supset}B \Rightarrow E \downarrow a$
using *provable.ImpLD*[**where** $\Gamma{=}\Gamma 1 \oplus C \oplus D{\supset}B \oplus D{\supset}B$ **and** $A{=}F$ **and** $B{=}H$ **and** $C{=}G$] **by** (*auto simp add*:*union-ac*)
then show *?case* **using** *eq2* **by** *blast*
next
case (*ImpLL* Γ F G H n E m Γ')
have $(F{\supset}G){\supset}H = (C{\supset}D){\supset}B \vee (F{\supset}G){\supset}H \neq (C{\supset}D){\supset}B$ **by** *blast*
moreover

```
    {assume (F⊃G)⊃H = (C⊃D)⊃B
     then have eqs: F=C G=D H=B Γ=Γ' using prems by auto
     from prems have Γ ⊕ H ⊕ F ⊕ G⊃H ⇒ E ↓ m using dpWeak' by auto
     moreover
     have Γ ⊕ F ⊕ G⊃H ⇒ G ↓ n by fact
     ultimately
      have ∃ n. Γ ⊕ F ⊕ G⊃H ⊕ G⊃H ⇒ E ↓ n using ImpLClassical[where Γ=Γ⊕F⊕G⊃H
and D=G and B=H and S=E and n=n]
        by (auto simp add:union-ac)
     then have ∃ n. Γ' ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ n using eqs by blast
    }
  moreover
    {assume (F⊃G)⊃H ≠ (C⊃D)⊃B
     then obtain Γ1 where eq1: Γ = Γ1 ⊕ (C⊃D)⊃B
                    and eq2: Γ' = Γ1 ⊕ (F⊃G)⊃H using ‹Γ ⊕ (F⊃G)⊃H = Γ' ⊕ (C⊃D)⊃B›
                                                midMultiset[where A=(F⊃G)⊃H and B=(C⊃D)⊃B]
by auto
     from eq1 have ∃ a. Γ1 ⊕ F ⊕ G⊃H ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ G ↓ a using prems(3)[where
Γ=Γ1 ⊕ F⊕G⊃H] by (auto simp add:union-ac)
     moreover
       from eq1 have ∃ a. Γ1 ⊕ H ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ a using prems(5)[where
Γ=Γ1⊕H] by (auto simp add:union-ac)
     ultimately
     have ∃ n. Γ1 ⊕ (F⊃G)⊃H ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ n using provable.ImpLL[where
Γ=Γ1⊕C⊕D⊃B⊕D⊃B and A=F and B=G and C=H and D=E]
       by (auto simp add:union-ac)
     then have ∃ n. Γ' ⊕ C ⊕ D⊃B ⊕ D⊃B ⇒ E ↓ n using eq2 by blast
    }
  ultimately
  show ?case by blast
qed


lemma contract:
fixes A :: form
 and n :: nat
assumes Γ ⊕ A ⊕ A ⇒ C ↓ n
shows ∃ k. Γ ⊕ A ⇒ C ↓ k
using assms
proof (induct x ≡ weight A y ≡ n  arbitrary: Γ A C n rule: nat-prod-induct)
  case (less x y)
  then have IH: ⋀ B m C Γ.⟦(weight B, m) <* (weight A, n); Γ ⊕ B ⊕ B ⇒ C ↓ m⟧ ⟹ ∃ a.
Γ ⊕ B ⇒ C ↓ a by simp
```

```
from prems show ?case
  proof (cases A)
    case ff
    then have Γ ⊕ A ⇒ C ↓ 0 by auto
    then show ∃ k. Γ ⊕ A ⇒ C ↓ k by blast
  next
    case (At i)
    then have Γ ⊕ At i ⊕ At i ⇒ C ↓ n using prems by auto
    then have ∃ k. Γ ⊕ At i ⇒ C ↓ k
       proof (cases)
          case (Ax j Γ')
          then have i=j ∨ i≠j by blast
          moreover
           {assume i=j
           then have Γ ⊕ At i ⇒ At i ↓ 0 by auto
           then have ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C = At j› and ‹i=j› by blast
           }
          moreover
           {assume i≠j
           then have At j :# Γ using ‹Γ ⊕ At i ⊕ At i = Γ'› and ‹At j :# Γ'› by auto
           then have Γ ⊕ At i ⇒ At j ↓ 0 by auto
           then have ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C = At j› by blast
           }
          ultimately
          show ∃ k. Γ ⊕ At i ⇒ C ↓ k by blast
       next
          case (LBot Γ')
          then have ff :# Γ by auto
          then show  ∃ k. Γ ⊕ At i ⇒ C ↓ k by (rule-tac x=0 in exI) (auto)
       next
          case (ConjR Γ' E s F t)
          then have Γ ⊕ At i ⊕ At i ⇒ E ↓ s by simp
          then have ∃ k. Γ ⊕ At i ⇒ E ↓ k
              using ‹A = At i› and ‹n=s+t+1›
                and IH[where B=At i and C=E and m=s] by auto
          moreover
          have Γ ⊕ At i ⊕ At i ⇒ F ↓ t using prems by simp
          then have ∃ k. Γ ⊕ At i ⇒ F ↓ k
              using ‹A = At i› and ‹n=s+t+1›
               and IH[where B=At i and C=F and m=t] by auto
          ultimately
          have ∃ k. Γ ⊕ At i ⇒ E∧*F ↓ k using provable.ConjR by auto
          then show ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C= E∧*F› by simp
```

next
case (*DisjR1* Γ' E s F)
then have $\Gamma \oplus At\ i \oplus At\ i \Rightarrow E \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow E \downarrow k$
using ‹$A = At\ i$› **and** ‹$n = s+1$›
and *IH*[**where** B=$At\ i$ **and** C=E **and** m=s] **by** *auto*
then have $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow E \vee * F \downarrow k$ **using** *provable.DisjR1* **by** *auto*
then show $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$ **using** ‹C=$E \vee * F$› **by** *simp*
next
case (*DisjR2* Γ' F s E)
then have $\Gamma \oplus At\ i \oplus At\ i \Rightarrow F \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow F \downarrow k$
using ‹$A = At\ i$› **and** ‹$n = s+1$›
and *IH*[**where** B=$At\ i$ **and** C=F **and** m=s] **by** *auto*
then have $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow E \vee * F \downarrow k$ **using** *provable.DisjR2* **by** *auto*
then show $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$ **using** ‹C=$E \vee * F$› **by** *simp*
next
case (*ImpR* Γ' E F s)
then have $\Gamma \oplus At\ i \oplus At\ i \oplus E \Rightarrow F \downarrow s$ **by** (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma \oplus At\ i \oplus E \Rightarrow F \downarrow k$
using ‹A=$At\ i$› **and** ‹n=$s+1$›
and *IH*[**where** B=$At\ i$ **and** C=F **and** m=s **and** Γ=$\Gamma \oplus E$]
by (*auto simp add*:*union-ac*)
then show $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$ **using** ‹C=$E \supset F$› **and** *provable.ImpR* **by** *auto*
next
case (*ConjL* Γ' E F D s)
from ‹$\Gamma \oplus At\ i \oplus At\ i = \Gamma' \oplus E \wedge * F$› **obtain** $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus E \wedge * F$
and *eq2*: $\Gamma' = \Gamma 1 \oplus At\ i \oplus At\ i$
using *midMultiset2*[**where** A=$At\ i$ **and** B=$E \wedge * F$] **by** *auto*
from *eq2* **and** ‹$\Gamma' \oplus E \oplus F \Rightarrow D \downarrow s$› **have** $\Gamma 1 \oplus At\ i \oplus At\ i \oplus E \oplus F \Rightarrow D \downarrow s$
by *simp*
then have $\exists\ k.\ \Gamma 1 \oplus At\ i \oplus E \oplus F \Rightarrow D \downarrow k$
using ‹A=$At\ i$› **and** ‹n=$s+1$› **and** *IH*[**where** Γ=$\Gamma 1 \oplus E \oplus F$ **and** B=$At\ i$] **by**
(*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma 1 \oplus At\ i \oplus E \wedge * F \Rightarrow D \downarrow k$ **using** *provable.ConjL* **by** *auto*
then show $\exists\ k.\ \Gamma \oplus At\ i \Rightarrow C \downarrow k$ **using** ‹C=D› **and** *eq1* **by** (*auto simp*
add:*union-ac*)
next
case (*DisjL* Γ' E D s F t)
from ‹$\Gamma \oplus At\ i \oplus At\ i = \Gamma' \oplus E \vee * F$› **obtain** $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus E \vee * F$
and *eq2*: $\Gamma' = \Gamma 1 \oplus At\ i \oplus At\ i$
using *midMultiset2*[**where** A=$At\ i$ **and** B=$E \vee * F$] **by** *auto*
from *eq2* **and** ‹$\Gamma' \oplus E \Rightarrow D \downarrow s$› **have** $\Gamma 1 \oplus At\ i \oplus At\ i \oplus E \Rightarrow D \downarrow s$ **by** *simp*

```
            then have ∃ k. Γ1 ⊕ At i ⊕ E ⇒ D ↓ k
                using ‹A=At i› and ‹n=s+t+1›
                and IH[where Γ=Γ1⊕E and m=s and B=At i] by (auto simp add:union-ac)
            moreover
            from eq2 and ‹Γ' ⊕ F ⇒ D ↓ t› have Γ1 ⊕ At i ⊕ At i ⊕ F ⇒ D ↓ t by simp
            then have ∃ k. Γ1 ⊕ At i ⊕ F ⇒ D ↓ k
                using ‹A=At i› and ‹n=s+t+1›
                and IH[where Γ=Γ1⊕F and m=t and B=At i] by (auto simp add:union-ac)
            ultimately
            have ∃ k. Γ1 ⊕ At i ⊕ E∨*F ⇒ D ↓ k using provable.DisjL[where Γ=Γ1⊕At i]
by auto
                then show ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp
add:union-ac)
        next
            case (ImpLC Γ' E F G D s)
            from ‹Γ ⊕ At i ⊕ At i = Γ' ⊕ (E∧*F)⊃G› obtain Γ1 where
                    eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
                    and eq2: Γ' = Γ1 ⊕ At i ⊕ At i
                using midMultiset2[where A=At i and B=(E∧*F)⊃G] by auto
            from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s› have Γ1 ⊕ At i ⊕ At i ⊕ E⊃(F⊃G) ⇒
D ↓ s by simp
            then have ∃ k. Γ1 ⊕ At i ⊕ E⊃(F⊃G) ⇒ D ↓ k
                using ‹A=At i› and ‹n=s+1›
                    and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=At i and m=s] by (auto simp
add:union-ac)
            then have ∃ k. Γ1 ⊕ At i ⊕ (E∧*F)⊃G ⇒ D ↓ k using provable.ImpLC[where
Γ=Γ1⊕At i] by auto
                then show ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp
add:union-ac)
        next
            case (ImpLD Γ' E G F D s)
            from ‹Γ ⊕ At i ⊕ At i = Γ' ⊕ (E∨*F)⊃G› obtain Γ1 where
                    eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
                    and eq2: Γ' = Γ1 ⊕ At i ⊕ At i
                using midMultiset2[where A=At i and B=(E∨*F)⊃G] by auto
            from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s› have Γ1 ⊕ At i ⊕ At i ⊕ E⊃G ⊕
F⊃G ⇒ D ↓ s by simp
            then have ∃ k. Γ1 ⊕ At i ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
                using ‹A=At i› and ‹n=s+1›
                    and IH[where Γ=Γ1⊕E⊃G⊕F⊃G and B=At i and m=s] by (auto simp
add:union-ac)
            then have ∃ k. Γ1 ⊕ At i ⊕ (E∨*F)⊃G ⇒ D ↓ k using provable.ImpLD[where
Γ=Γ1⊕At i] by auto
```

```
                then show ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp
add:union-ac)
        next
          case (ImpLL Γ' E F G s D t)
          from ‹Γ ⊕ At i ⊕ At i = Γ' ⊕ (E⊃F)⊃G› obtain Γ1 where
                 eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
                and eq2: Γ' = Γ1 ⊕ At i ⊕ At i
            using midMultiset2[where A=At i and B=(E⊃F)⊃G] by auto
          from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s› have Γ1 ⊕ At i ⊕ At i ⊕ E ⊕ F⊃G ⇒
F ↓ s by simp
          then have ∃ k. Γ1 ⊕ At i ⊕ E ⊕ F⊃G ⇒ F ↓ k
            using ‹A=At i› and ‹n=s+t+1›
                 and IH[where Γ=Γ1⊕E⊕F⊃G and B=At i and m=s] by (auto simp
add:union-ac)
          moreover
          from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t› have Γ1 ⊕ At i ⊕ At i ⊕ G ⇒ D ↓ t by simp
          then have ∃ k. Γ1 ⊕ At i ⊕ G ⇒ D ↓ k
            using ‹A=At i› and ‹n=s+t+1›
            and IH[where Γ=Γ1⊕G and B=At i and m=t] by (auto simp add:union-ac)
          ultimately
         have ∃ k. Γ1 ⊕ At i ⊕ (E⊃F)⊃G ⇒ D ↓ k using provable.ImpLL[where Γ=Γ⊕At
i] by auto
                then show ∃ k. Γ ⊕ At i ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp
add:union-ac)
        next
          case (ImpL0 Γ' j E D s)
          have i=j ∨ i≠j by blast
          moreover
            {assume i=j
           then have Γ ⊕ At i ⊕ At i = Γ' ⊕ At i ⊃ E ⊕ At i using prems by (auto simp
add:union-ac)
            then have Γ ⊕ At i = Γ' ⊕ At i ⊃ E by simp
            then obtain Γ1 where eq1: Γ = Γ1 ⊕ At i ⊃ E
                             and eq2: Γ' = Γ1 ⊕ At i
              using midMultiset[where A=At i and B=At i ⊃ E] by auto
            from eq2 and ‹Γ' ⊕ At j ⊕ E ⇒ D ↓ s› and ‹i=j›
              have Γ1 ⊕ At i ⊕ At i ⊕ E ⇒ D ↓ s by simp
            then have ∃ k. Γ1 ⊕ At i ⊕ E ⇒ D ↓ k
              using ‹A = At i› and ‹n=s+1›
               and IH[where B=At i and Γ=Γ1⊕E] by (auto simp add:union-ac)
           then have ∃ k. Γ1 ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k using provable.ImpL0 by auto
          then have ∃ k. Γ ⊕ At i ⇒ C ↓ k using eq1 and ‹C=D› by (simp add:union-ac)
            }
```

```
            moreover
              {assume i≠j
               from ‹Γ ⊕ At i ⊕ At i = Γ′ ⊕ At j ⊕ At j ⊃ E› obtain Γ2 where
                      eq1: Γ = Γ2 ⊕ At j ⊃ E
                    and eq2: Γ′ ⊕ At j = Γ2 ⊕ At i ⊕ At i
                  using midMultiset2[where A=At i and B=At j⊃E] by auto
               from eq2 and ‹i≠j› obtain Γ1 where eq3: Γ′ = Γ1 ⊕ At i ⊕ At i
                                              and eq4: Γ2 = Γ1 ⊕ At j
                  using midMultiset2[where A=At i and B=At j and Γ=Γ2 and Γ′=Γ′] by
auto
               from eq4 and eq1 have eq: Γ = Γ1 ⊕ At j ⊕ At j ⊃ E by auto
               from eq3 and ‹Γ′ ⊕ At j ⊕ E ⇒ D ↓ s›
                  have Γ1 ⊕ At i ⊕ At i ⊕ At j ⊕ E ⇒ D ↓ s by simp
               then have ∃ k. Γ1 ⊕ At i ⊕ At j ⊕ E ⇒ D ↓ k
                  using ‹A=At i› and ‹n=s+1›
                    and IH[where Γ=Γ1⊕At j ⊕ E and B=At i and C=D] by (auto simp
add:union-ac)
               then have ∃ k. Γ1 ⊕ At i ⊕ At j ⊕ At j ⊃ E ⇒ D ↓ k
               using provable.ImpL0[where Γ=Γ1⊕At i and i=j] by (auto simp add:union-ac)
                  then have ∃ k. Γ ⊕ At i ⇒ C ↓ k using eq and ‹C=D› by (auto simp
add:union-ac)
              }
            ultimately
            show ∃ k. Γ ⊕ At i ⇒ C ↓ k by blast
          qed
          then show ∃ k. Γ ⊕ A ⇒ C ↓ k using ‹A = At i› by simp
      next
        case (Conj S T)
        then have Γ ⊕ S∧∗T ⊕ S∧∗T ⇒ C ↓ n using prems by auto
        then have ∃ k. Γ ⊕ S∧∗T ⇒ C ↓ k
          proof (cases)
            case (Ax i Γ′)
            then have At i :# Γ by auto
            then have Γ ⊕ S∧∗T ⇒ At i ↓ 0 by auto
            then show ∃ k. Γ ⊕ S∧∗T ⇒ C ↓ k using ‹C=At i› by blast
          next
            case (LBot Γ′ D)
            then have ff :# Γ by auto
            then have Γ ⊕ S∧∗T ⇒ D ↓ 0 by auto
            then show ∃ k. Γ ⊕ S∧∗T ⇒ C ↓ k using ‹C=D› by auto
          next
            case (DisjR1 Γ′ E s F)
            then have Γ ⊕ S∧∗T ⊕ S∧∗T ⇒ E ↓ s by simp
```

```
            then have ∃ k. Γ ⊕ S∧*T ⇒ E ↓ k
              using ‹A = S∧*T› and ‹n = s+1›
              and IH[where B=S∧*T and C=E and m=s] by auto
            then have ∃ k. Γ ⊕ S∧*T ⇒ E∨*F ↓ k using provable.DisjR1 by auto
            then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=E∨*F› by simp
          next
            case (DisjR2 Γ' F s E)
            then have Γ ⊕ S∧*T ⊕ S∧*T ⇒ F ↓ s by simp
            then have ∃ k. Γ ⊕ S∧*T ⇒ F ↓ k
              using ‹A = S∧*T› and ‹n = s+1› and IH[where B=S∧*T and C=F and
m=s] by auto
            then have ∃ k. Γ ⊕ S∧*T ⇒ E∨*F ↓ k using provable.DisjR2 by auto
            then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=E∨*F› by simp
          next
            case (ImpR Γ' E F s)
            then have Γ ⊕ S∧*T ⊕ S∧*T ⊕ E ⇒ F ↓ s by (auto simp add:union-ac)
            then have ∃ k. Γ ⊕ S∧*T ⊕ E ⇒ F ↓ k
              using ‹A=S∧*T› and ‹n=s+1›
              and IH[where B=S∧*T and C=F and m=s and Γ=Γ⊕E] by (auto simp
add:union-ac)
            then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=E⊃F› and provable.ImpR by auto
          next
            case (ConjR Γ' E s F t)
            then have Γ ⊕ S∧*T ⊕ S∧*T ⇒ E ↓ s by simp
            then have ∃ k. Γ ⊕ S∧*T ⇒ E ↓ k
              using ‹A = S∧*T› and ‹n=s+t+1› and IH[where B=S∧*T and C=E and
m=s] by auto
            moreover
            have Γ ⊕ S∧*T ⊕ S∧*T ⇒ F ↓ t using prems by simp
            then have ∃ k. Γ ⊕ S∧*T ⇒ F ↓ k
              using ‹A = S∧*T› and ‹n=s+t+1› and IH[where B=S∧*T and C=F and
m=t] by auto
            ultimately
            have ∃ k. Γ ⊕ S∧*T ⇒ E∧*F ↓ k using provable.ConjR by auto
            then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C= E∧*F› by simp
          next
            case (DisjL Γ' E D s F t)
            from ‹Γ ⊕ S∧*T ⊕ S∧*T = Γ' ⊕ E∨*F› obtain Γ1 where
              eq1: Γ = Γ1 ⊕ E∨*F
              and eq2: Γ' = Γ1 ⊕ S∧*T ⊕ S∧*T
              using midMultiset2[where A=S∧*T and B=E∨*F] by auto
            from eq2 and ‹Γ' ⊕ E ⇒ D ↓ s› have Γ1 ⊕ S∧*T ⊕ S∧*T ⊕ E ⇒ D ↓ s by simp
            then have ∃ k. Γ1 ⊕ S∧*T ⊕ E ⇒ D ↓ k
```

```
            using ‹A=S∧*T› and ‹n=s+t+1›
           and IH[where Γ=Γ1⊕E and m=s and B=S∧*T] by (auto simp add:union-ac)
          moreover
         from eq2 and ‹Γ' ⊕ F ⇒ D ↓ t› have Γ1 ⊕ S∧*T ⊕ S∧*T ⊕ F ⇒ D ↓ t by simp
          then have ∃ k. Γ1 ⊕ S∧*T ⊕ F ⇒ D ↓ k
            using ‹A=S∧*T› and ‹n=s+t+1›
           and IH[where Γ=Γ1⊕F and m=t and B=S∧*T] by (auto simp add:union-ac)
         ultimately
        have ∃ k. Γ1 ⊕ S∧*T ⊕ E∨*F ⇒ D ↓ k using provable.DisjL[where Γ=Γ1⊕S∧*T]
by auto
            then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp
add:union-ac)
      next
        case (ImpLC Γ' E F G D s)
        from ‹Γ ⊕ S∧*T ⊕ S∧*T = Γ' ⊕ (E∧*F)⊃G› obtain Γ1 where
            eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
            and eq2: Γ' = Γ1 ⊕ S∧*T ⊕ S∧*T
           using midMultiset2[where A=S∧*T and B=(E∧*F)⊃G] by auto
        from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s›
            have Γ1 ⊕ S∧*T ⊕ S∧*T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
        then have ∃ k. Γ1 ⊕ S∧*T ⊕ E⊃(F⊃G) ⇒ D ↓ k
            using ‹A=S∧*T› and ‹n=s+1›
              and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S∧*T and m=s] by (auto simp
add:union-ac)
        then have ∃ k. Γ1 ⊕ S∧*T ⊕ (E∧*F)⊃G ⇒ D ↓ k
            using provable.ImpLC[where Γ=Γ1⊕S∧*T] by auto
       then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
      next
        case (ImpLD Γ' E G F D s)
        from ‹Γ ⊕ S∧*T ⊕ S∧*T = Γ' ⊕ (E∨*F)⊃G› obtain Γ1 where
            eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
            and eq2: Γ' = Γ1 ⊕ S∧*T ⊕ S∧*T
            using midMultiset2[where A=S∧*T and B=(E∨*F)⊃G] by auto
       from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
           have Γ1 ⊕ S∧*T ⊕ S∧*T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
       then have ∃ k. Γ1 ⊕ S∧*T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
            using ‹A=S∧*T› and ‹n=s+1›
              and IH[where Γ=Γ1⊕E⊃G⊕F⊃G and B=S∧*T and m=s] by (auto simp
add:union-ac)
         then have ∃ k. Γ1 ⊕ S∧*T ⊕ (E∨*F)⊃G ⇒ D ↓ k
           using provable.ImpLD[where Γ=Γ1⊕S∧*T] by auto
       then show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
      next
```

case (*ImpLL* Γ' E F G s D t)
from ‹$\Gamma \oplus S \wedge\!* T \oplus S \wedge\!* T = \Gamma' \oplus (E \supset F) \supset G$› **obtain** $\Gamma 1$ **where**
eq1: $\Gamma = \Gamma 1 \oplus (E \supset F) \supset G$
and *eq2*: $\Gamma' = \Gamma 1 \oplus S \wedge\!* T \oplus S \wedge\!* T$
using *midMultiset2*[**where** $A{=}S \wedge\!* T$ **and** $B{=}(E \supset F) \supset G$] **by** *auto*
from *eq2* **and** ‹$\Gamma' \oplus E \oplus F \supset G \Rightarrow F \downarrow s$›
have $\Gamma 1 \oplus S \wedge\!* T \oplus S \wedge\!* T \oplus E \oplus F \supset G \Rightarrow F \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus S \wedge\!* T \oplus E \oplus F \supset G \Rightarrow F \downarrow k$
using ‹$A{=}S \wedge\!* T$› **and** ‹$n{=}s{+}t{+}1$›
and *IH*[**where** $\Gamma{=}\Gamma 1 \oplus E \oplus F \supset G$ **and** $B{=}S \wedge\!* T$ **and** $m{=}s$] **by** (*auto simp add*:*union-ac*)
moreover
from *eq2* **and** ‹$\Gamma' \oplus G \Rightarrow D \downarrow t$› **have** $\Gamma 1 \oplus S \wedge\!* T \oplus S \wedge\!* T \oplus G \Rightarrow D \downarrow t$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus S \wedge\!* T \oplus G \Rightarrow D \downarrow k$
using ‹$A{=}S \wedge\!* T$› **and** ‹$n{=}s{+}t{+}1$›
and *IH*[**where** $\Gamma{=}\Gamma 1 \oplus G$ **and** $B{=}S \wedge\!* T$ **and** $m{=}t$] **by** (*auto simp add*:*union-ac*)
ultimately
have $\exists\ k.\ \Gamma 1 \oplus S \wedge\!* T \oplus (E \supset F) \supset G \Rightarrow D \downarrow k$
using *provable.ImpLL*[**where** $\Gamma{=}\Gamma \oplus S \wedge\!* T$] **by** *auto*
then show $\exists\ k.\ \Gamma \oplus S \wedge\!* T \Rightarrow C \downarrow k$ **using** ‹$C{=}D$› **and** *eq1* **by** (*auto simp add*:*union-ac*)
next
case (*ImpL0* Γ' j E D s)
from ‹$\Gamma \oplus S \wedge\!* T \oplus S \wedge\!* T = \Gamma' \oplus At\ j \oplus At\ j \supset E$› **obtain** $\Gamma 2$ **where**
eq1: $\Gamma = \Gamma 2 \oplus At\ j \supset E$
and *eq2*: $\Gamma' \oplus At\ j = \Gamma 2 \oplus S \wedge\!* T \oplus S \wedge\!* T$
using *midMultiset2*[**where** $A{=}S \wedge\!* T$ **and** $B{=}At\ j \supset E$] **by** *auto*
from *eq2* **obtain** $\Gamma 1$ **where** *eq3*: $\Gamma' = \Gamma 1 \oplus S \wedge\!* T \oplus S \wedge\!* T$
and *eq4*: $\Gamma 2 = \Gamma 1 \oplus At\ j$
using *midMultiset2*[**where** $A{=}S \wedge\!* T$ **and** $B{=}At\ j$ **and** $\Gamma{=}\Gamma 2$ **and** $\Gamma'{=}\Gamma'$] **by** *auto*
from *eq4* **and** *eq1* **have** *eq*: $\Gamma = \Gamma 1 \oplus At\ j \oplus At\ j \supset E$ **by** *auto*
from *eq3* **and** ‹$\Gamma' \oplus At\ j \oplus E \Rightarrow D \downarrow s$›
have $\Gamma 1 \oplus S \wedge\!* T \oplus S \wedge\!* T \oplus At\ j \oplus E \Rightarrow D \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus S \wedge\!* T \oplus At\ j \oplus E \Rightarrow D \downarrow k$
using ‹$A{=}S \wedge\!* T$› **and** ‹$n{=}s{+}1$›
and *IH*[**where** $\Gamma{=}\Gamma 1 \oplus At\ j \oplus E$ **and** $B{=}S \wedge\!* T$ **and** $C{=}D$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma 1 \oplus S \wedge\!* T \oplus At\ j \oplus At\ j \supset E \Rightarrow D \downarrow k$
using *provable.ImpL0*[**where** $\Gamma{=}\Gamma 1 \oplus S \wedge\!* T$ **and** $i{=}j$] **by** (*auto simp add*:*union-ac*)
then show $\exists\ k.\ \Gamma \oplus S \wedge\!* T \Rightarrow C \downarrow k$ **using** *eq* **and** ‹$C{=}D$› **by** (*auto simp add*:*union-ac*)
next
case (*ConjL* Γ' E F D s)
have $S \wedge\!* T = E \wedge\!* F \vee S \wedge\!* T \neq E \wedge\!* F$ **by** *blast*
moreover
{assume $S \wedge\!* T = E \wedge\!* F$

```
          then have Γ ⊕ E∧*F = Γ' using ‹Γ ⊕ S ∧* T ⊕ S ∧* T = Γ' ⊕ E ∧* F› by simp
          then have Γ ⊕ E∧*F ⊕ E ⊕ F ⇒ D ↓ s using ‹Γ' ⊕ E ⊕ F ⇒ D ↓ s› by simp
           then have ∃ k. Γ ⊕ E ⊕ F ⊕ E ⊕ F ⇒ D ↓ k
              using inversionConjL[where Γ=Γ⊕E⊕F and A=E and B=F]
              by (auto simp add:union-ac)
          then have ∃ k. Γ ⊕ E ⊕ F ⊕ F ⇒ D ↓ k using ‹S∧*T = E∧*F› and ‹A=S∧*T›
               and IH[where B=E and Γ=Γ⊕F⊕F and C=D] by (auto simp add:union-ac)
         then have ∃ k. Γ ⊕ E ⊕ F ⇒ D ↓ k using ‹S∧*T = E∧*F› and ‹A=S∧*T›
             and IH[where B=F and Γ=Γ⊕E and C=D] by auto
         then have ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹S∧*T = E∧*F›
             and ‹C=D› and provable.ConjL by auto
          }
        moreover
          {assume S∧*T ≠ E∧*F
           from ‹Γ ⊕ S∧*T ⊕ S∧*T = Γ' ⊕ E∧*F› obtain Γ1 where eq1: Γ = Γ1 ⊕ E∧*F
                    and eq2: Γ' = Γ1 ⊕ S∧*T ⊕ S∧*T
              using midMultiset2[where A=S∧*T and B=E∧*F] and ‹S∧*T ≠ E∧*F› by
auto
           from eq2 and ‹Γ' ⊕ E ⊕ F ⇒ D ↓ s› have Γ1 ⊕ S∧*T ⊕ S∧*T ⊕ E ⊕ F ⇒ D ↓
s by simp
           then have ∃ k. Γ1 ⊕ S∧*T ⊕ E ⊕ F ⇒ D ↓ k
              using ‹A=S∧*T› and ‹n=s+1›
              and IH[where Γ=Γ1⊕E⊕F and B=S∧*T] by (auto simp add:union-ac)
           then have ∃ k. Γ1 ⊕ S∧*T ⊕ E∧*F ⇒ D ↓ k using provable.ConjL by auto
         then have ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
          }
        ultimately
        show ∃ k. Γ ⊕ S∧*T ⇒ C ↓ k by blast
      qed
    then show ∃ k. Γ ⊕ A ⇒ C ↓ k using ‹A = S∧*T› by simp
      next
       case (Disj S T)
       then have Γ ⊕ S∨*T ⊕ S∨*T ⇒ C ↓ n using prems by auto
       then have ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k
         proof (cases)
         case (Ax i Γ')
         then have At i :# Γ by auto
         then have Γ ⊕ S∨*T ⇒ At i ↓ 0 by auto
         then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=At i› by blast
       next
         case (LBot Γ' D)
       then have ff :# Γ by auto
        then have Γ ⊕ S∨*T ⇒ D ↓ 0 by auto
```

```
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=D› by auto
    next
      case (DisjR1 Γ' E s F)
      then have Γ ⊕ S∨*T ⊕ S∨*T ⇒ E ↓ s by simp
      then have ∃ k. Γ ⊕ S∨*T ⇒ E ↓ k
        using ‹A = S∨*T› and ‹n = s+1›
        and IH[where B=S∨*T and C=E and m=s] by auto
      then have ∃ k. Γ ⊕ S∨*T ⇒ E∨*F ↓ k using provable.DisjR1 by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=E∨*F› by simp
    next
      case (DisjR2 Γ' F s E)
      then have Γ ⊕ S∨*T ⊕ S∨*T ⇒ F ↓ s by simp
      then have ∃ k. Γ ⊕ S∨*T ⇒ F ↓ k
          using ‹A = S∨*T› and ‹n = s+1›
          and IH[where B=S∨*T and C=F and m=s] by auto
      then have ∃ k. Γ ⊕ S∨*T ⇒ E∨*F ↓ k using provable.DisjR2 by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=E∨*F› by simp
    next
      case (ImpR Γ' E F s)
      then have Γ ⊕ S∨*T ⊕ S∨*T ⊕ E ⇒ F ↓ s by (auto simp add:union-ac)
      then have ∃ k. Γ ⊕ S∨*T ⊕ E ⇒ F ↓ k
        using ‹A=S∨*T› and ‹n=s+1›
            and IH[where B=S∨*T and C=F and m=s and Γ=Γ⊕E] by (auto simp
add:union-ac)
       then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=E⊃F› and provable.ImpR by auto
     next
      case (ConjR Γ' E s F t)
      then have Γ ⊕ S∨*T ⊕ S∨*T ⇒ E ↓ s by simp
      then have ∃ k. Γ ⊕ S∨*T ⇒ E ↓ k
          using ‹A = S∨*T› and ‹n=s+t+1›
          and IH[where B=S∨*T and C=E and m=s] by auto
      moreover
      have Γ ⊕ S∨*T ⊕ S∨*T ⇒ F ↓ t using prems by simp
      then have ∃ k. Γ ⊕ S∨*T ⇒ F ↓ k
          using ‹A = S∨*T› and ‹n=s+t+1›
          and IH[where B=S∨*T and C=F and m=t] by auto
      ultimately
      have ∃ k. Γ ⊕ S∨*T ⇒ E∧*F ↓ k using provable.ConjR by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C= E∧*F› by simp
    next
      case (ConjL Γ' E F D s)
     from ‹Γ ⊕ S∨*T ⊕ S∨*T = Γ' ⊕ E∧*F› obtain Γ1 where
          eq1: Γ = Γ1 ⊕ E∧*F
```

```
            and eq2: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
            using midMultiset2[where A=S∨*T and B=E∧*F] by auto
        from eq2 and ⟨Γ' ⊕ E ⊕ F ⇒ D ↓ s⟩
            have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ E ⊕ F ⇒ D ↓ s by simp
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ E ⊕ F ⇒ D ↓ k
          using ⟨A=S∨*T⟩ and ⟨n=s+1⟩
            and IH[where Γ=Γ1⊕E⊕F and B=S∨*T] by (auto simp add:union-ac)
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ E∧*F ⇒ D ↓ k using provable.ConjL by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ⟨C=D⟩ and eq1 by (auto simp add:union-ac)
    next
      case (ImpLC Γ' E F G D s)
      from ⟨Γ ⊕ S∨*T ⊕ S∨*T = Γ' ⊕ (E∧*F)⊃G⟩ obtain Γ1 where
          eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
          and eq2: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
          using midMultiset2[where A=S∨*T and B=(E∧*F)⊃G] by auto
      from eq2 and ⟨Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s⟩ have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ E⊃(F⊃G) ⇒
D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S∨*T ⊕ E⊃(F⊃G) ⇒ D ↓ k
          using ⟨A=S∨*T⟩ and ⟨n=s+1⟩ and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S∨*T
and m=s] by (auto simp add:union-ac)
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ (E∧*F)⊃G ⇒ D ↓ k using provable.ImpLC[where
Γ=Γ1⊕S∨*T] by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ⟨C=D⟩ and eq1 by (auto simp add:union-ac)
    next
      case (ImpLD Γ' E G F D s)
      from ⟨Γ ⊕ S∨*T ⊕ S∨*T = Γ' ⊕ (E∨*F)⊃G⟩ obtain Γ1 where
          eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
          and eq2: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
          using midMultiset2[where A=S∨*T and B=(E∨*F)⊃G] by auto
      from eq2 and ⟨Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s⟩
          have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S∨*T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
          using ⟨A=S∨*T⟩ and ⟨n=s+1⟩
              and IH[where Γ=Γ1⊕E⊃G⊕F⊃G and B=S∨*T and m=s] by (auto simp
add:union-ac)
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ (E∨*F)⊃G ⇒ D ↓ k using provable.ImpLD[where
Γ=Γ1⊕S∨*T] by auto
      then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ⟨C=D⟩ and eq1 by (auto simp add:union-ac)
    next
      case (ImpLL Γ' E F G s D t)
      from ⟨Γ ⊕ S∨*T ⊕ S∨*T = Γ' ⊕ (E⊃F)⊃G⟩ obtain Γ1 where
          eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
          and eq2: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
```

```
            using midMultiset2[where A=S∨*T and B=(E⊃F)⊃G] by auto
        from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s›
            have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ E ⊕ F⊃G ⇒ F ↓ k
            using ‹A=S∨*T› and ‹n=s+t+1›
          and IH[where Γ=Γ1⊕E⊕F⊃G and B=S∨*T and m=s] by (auto simp add:union-ac)
        moreover
        from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t› have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ G ⇒ D ↓ t by simp
        then have ∃ k. Γ1 ⊕ S∨*T ⊕ G ⇒ D ↓ k
            using ‹A=S∨*T› and ‹n=s+t+1›
            and IH[where Γ=Γ1⊕G and B=S∨*T and m=t] by (auto simp add:union-ac)
        ultimately
       have ∃ k. Γ1 ⊕ S∨*T ⊕ (E⊃F)⊃G ⇒ D ↓ k using provable.ImpLL[where Γ=Γ⊕S∨*T]
by auto
       then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
    next
       case (ImpL0 Γ' j E D s)
       from ‹Γ ⊕S∨*T ⊕ S∨*T = Γ' ⊕ At j ⊕ At j ⊃ E› obtain Γ2 where
            eq1: Γ = Γ2 ⊕ At j ⊃ E
            and eq2: Γ' ⊕ At j = Γ2 ⊕ S∨*T ⊕ S∨*T
            using midMultiset2[where A=S∨*T and B=At j⊃E] by auto
       from eq2 obtain Γ1 where eq3: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
            and eq4: Γ2 = Γ1 ⊕ At j
            using midMultiset2[where A=S∨*T and B=At j and Γ=Γ2 and Γ'=Γ'] by auto
       from eq4 and eq1 have eq: Γ = Γ1 ⊕ At j ⊕ At j ⊃ E by auto
      from eq3 and ‹Γ' ⊕ At j ⊕ E ⇒ D ↓ s›
           have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ At j ⊕ E ⇒ D ↓ s by simp
       then have ∃ k. Γ1 ⊕ S∨*T ⊕ At j ⊕ E ⇒ D ↓ k
           using ‹A=S∨*T› and ‹n=s+1›
         and IH[where Γ=Γ1⊕At j ⊕ E and B=S∨*T and C=D] by (auto simp add:union-ac)
       then have ∃ k. Γ1 ⊕ S∨*T ⊕ At j ⊕ At j ⊃ E ⇒ D ↓ k
            using provable.ImpL0[where Γ=Γ1⊕S∨*T and i=j] by (auto simp add:union-ac)
        then show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using eq and ‹C=D› by (auto simp add:union-ac)
    next
       case (DisjL Γ' E D s F t)
       have S∨*T = E∨*F ∨ S∨*T ≠ E∨*F by blast
       moreover
        {assume S∨*T = E∨*F
        then have Γ ⊕ E∨*F = Γ' using ‹Γ ⊕S∨*T⊕S∨*T = Γ' ⊕ E∨*F› by simp
        from ‹Γ' ⊕ E ⇒ D ↓ s› and ‹Γ ⊕ E∨*F = Γ'›
            have Γ ⊕ E∨*F ⊕ E ⇒ D ↓ s by simp
         then have ∃ k. Γ ⊕ E ⊕ E ⇒ D ↓ k using inversionDisjL[where Γ=Γ⊕E and A=E
and B=F]
```

```
        by (auto simp add:union-ac)
      then have ∃ k. Γ ⊕ E ⇒ D ↓ k using ‹S∨*T=E∨*F› and ‹A=S∨*T›
        and IH by auto
      moreover
      from ‹Γ' ⊕ F ⇒ D ↓ t› and ‹Γ ⊕ E∨*F = Γ'›
        have Γ ⊕ E∨*F ⊕ F ⇒ D ↓ t by simp
      then have ∃ k. Γ ⊕ F ⊕ F ⇒ D ↓ k using inversionDisjL[where Γ=Γ⊕F and A=E
and B=F]
        by (auto simp add:union-ac)
      then have ∃ k. Γ ⊕ F ⇒ D ↓ k using ‹S∨*T=E∨*F› and ‹A=S∨*T›
        and IH by auto
      ultimately
      have ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹S∨*T=E∨*F› and ‹C=D› and provable.DisjL
by auto
      }
    moreover
      {assume S∨*T≠E∨*F
      from ‹Γ ⊕ S∨*T ⊕ S∨*T = Γ' ⊕ E∨*F› obtain Γ1 where
        eq1: Γ = Γ1 ⊕ E∨*F
        and eq2: Γ' = Γ1 ⊕ S∨*T ⊕ S∨*T
        using midMultiset2[where A=S∨*T and B=E∨*F] and ‹S∨*T≠E∨*F› by auto
      from eq2 and ‹Γ' ⊕ E ⇒ D ↓ s› have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ E ⇒ D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S∨*T ⊕ E ⇒ D ↓ k
        using ‹A=S∨*T› and ‹n=s+t+1›
        and IH[where Γ=Γ1⊕E and m=s and B=S∨*T] by (auto simp add:union-ac)
      moreover
      from eq2 and ‹Γ' ⊕ F ⇒ D ↓ t› have Γ1 ⊕ S∨*T ⊕ S∨*T ⊕ F ⇒ D ↓ t by simp
      then have ∃ k. Γ1 ⊕ S∨*T ⊕ F ⇒ D ↓ k
        using ‹A=S∨*T› and ‹n=s+t+1›
        and IH[where Γ=Γ1⊕F and m=t and B=S∨*T] by (auto simp add:union-ac)
      ultimately
      have ∃ k. Γ1 ⊕ S∨*T ⊕ E∨*F ⇒ D ↓ k using provable.DisjL[where Γ=Γ1⊕S∨*T]
by auto
      then have ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
      }
    ultimately
    show ∃ k. Γ ⊕ S∨*T ⇒ C ↓ k by blast
    qed
  then show ∃ k. Γ ⊕ A ⇒ C ↓ k using ‹A = S∨*T› by simp
next
  case (Imp S T)
  then have Γ ⊕ S⊃T ⊕ S⊃T ⇒ C ↓ n using prems by auto
  then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
```

proof (*cases*)
  case (*Ax i* Γ')
  then have *At i* :# Γ **by** *auto*
  then have $\Gamma \oplus S \supset T \Rightarrow$ *At i* $\downarrow$ *0* **by** *auto*
  then show $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow C \downarrow k$ **using** ‹*C*=*At i*› **by** *blast*
next
  case (*LBot* Γ' *D*)
  then have *ff* :# Γ **by** *auto*
  then have $\Gamma \oplus S \supset T \Rightarrow D \downarrow 0$ **by** *auto*
  then show $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow C \downarrow k$ **using** ‹*C*=*D*› **by** *auto*
next
  case (*DisjR1* Γ' *E s F*)
  then have $\Gamma \oplus S \supset T \oplus S \supset T \Rightarrow E \downarrow s$ **by** *simp*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow E \downarrow k$
    using ‹$A = S \supset T$› **and** ‹$n = s+1$›
    and *IH*[**where** $B{=}S \supset T$ **and** $C{=}E$ **and** $m{=}s$] **by** *auto*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow E \vee * F \downarrow k$ **using** *provable.DisjR1* **by** *auto*
  then show $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow C \downarrow k$ **using** ‹$C{=}E \vee * F$› **by** *simp*
next
  case (*DisjR2* Γ' *F s E*)
  then have $\Gamma \oplus S \supset T \oplus S \supset T \Rightarrow F \downarrow s$ **by** *simp*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow F \downarrow k$
    using ‹$A = S \supset T$› **and** ‹$n = s+1$›
    and *IH*[**where** $B{=}S \supset T$ **and** $C{=}F$ **and** $m{=}s$] **by** *auto*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow E \vee * F \downarrow k$ **using** *provable.DisjR2* **by** *auto*
  then show $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow C \downarrow k$ **using** ‹$C{=}E \vee * F$› **by** *simp*
next
  case (*ImpR* Γ' *E F s*)
  then have $\Gamma \oplus S \supset T \oplus S \supset T \oplus E \Rightarrow F \downarrow s$ **by** (*auto simp add*:*union-ac*)
  then have $\exists\ k.\ \Gamma \oplus S \supset T \oplus E \Rightarrow F \downarrow k$
    using ‹$A{=}S \supset T$› **and** ‹$n{=}s+1$›
   and *IH*[**where** $B{=}S \supset T$ **and** $C{=}F$ **and** $m{=}s$ **and** $\Gamma{=}\Gamma \oplus E$] **by** (*auto simp add*:*union-ac*)

  then show $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow C \downarrow k$ **using** ‹$C{=}E \supset F$› **and** *provable.ImpR* **by** *auto*
next
  case (*ConjR* Γ' *E s F t*)
  then have $\Gamma \oplus S \supset T \oplus S \supset T \Rightarrow E \downarrow s$ **by** *simp*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow E \downarrow k$
    using ‹$A = S \supset T$› **and** ‹$n{=}s+t+1$›
    and *IH*[**where** $B{=}S \supset T$ **and** $C{=}E$ **and** $m{=}s$] **by** *auto*
  moreover
  have $\Gamma \oplus S \supset T \oplus S \supset T \Rightarrow F \downarrow t$ **using** *prems* **by** *simp*
  then have $\exists\ k.\ \Gamma \oplus S \supset T \Rightarrow F \downarrow k$

```
        using ‹A = S⊃T› and ‹n=s+t+1›
         and IH[where B=S⊃T and C=F and m=t] by auto
      ultimately
      have ∃ k. Γ ⊕ S⊃T ⇒ E∧*F ↓ k using provable.ConjR by auto
      then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ‹C= E∧*F› by simp
    next
      case (ConjL Γ' E F D s)
      from ‹Γ ⊕ S⊃T ⊕ S⊃T = Γ' ⊕ E∧*F› obtain Γ1 where
          eq1: Γ = Γ1 ⊕ E∧*F
           and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
          using midMultiset2[where A=S⊃T and B=E∧*F] by auto
      from eq2 and ‹Γ' ⊕ E ⊕ F ⇒ D ↓ s›
          have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F ⇒ D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F ⇒ D ↓ k
          using ‹A=S⊃T› and ‹n=s+1›
           and IH[where Γ=Γ1⊕E⊕F and B=S⊃T] by (auto simp add:union-ac)
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ E∧*F ⇒ D ↓ k using provable.ConjL by auto
      then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ‹C=D› and eq1 by (auto simp add:union-ac)
    next
      case (DisjL Γ' E D s F t)
      from ‹Γ ⊕ S⊃T ⊕ S⊃T = Γ' ⊕ E∨*F› obtain Γ1 where
          eq1: Γ = Γ1 ⊕ E∨*F
           and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
          using midMultiset2[where A=S⊃T and B=E∨*F] by auto
      from eq2 and ‹Γ' ⊕ E ⇒ D ↓ s›
          have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⇒ D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⇒ D ↓ k
          using ‹A=S⊃T› and ‹n=s+t+1›
          and IH[where Γ=Γ1⊕E and m=s and B=S⊃T]
          by (auto simp add:union-ac)
      moreover
      from eq2 and ‹Γ' ⊕ F ⇒ D ↓ t›
          have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ F ⇒ D ↓ t by simp
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ F ⇒ D ↓ k
          using ‹A=S⊃T› and ‹n=s+t+1›
          and IH[where Γ=Γ1⊕F and m=t and B=S⊃T]
          by (auto simp add:union-ac)
      ultimately
       have ∃ k. Γ1 ⊕ S⊃T ⊕ E∨*F ⇒ D ↓ k using provable.DisjL[where Γ=Γ1⊕S⊃T] by
auto
      then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ‹C=D› and eq1
          by (auto simp add:union-ac)
    next
```

case (*ImpL0* Γ' *i E D s*)
then have $\exists\ k.\ \Gamma \oplus S{\supset}T \Rightarrow C \downarrow k$
proof (*cases S*)
case (*At j*)
have $At\ j \supset T = At\ i \supset E \vee At\ j \supset T \neq At\ i \supset E$ **by** *blast*
moreover
{**assume** $At\ j \supset T = At\ i \supset E$
then have $\Gamma \oplus At\ i \supset E\ = \Gamma' \oplus At\ i$
using ‹$\Gamma \oplus S{\supset}T \oplus S{\supset}T = \Gamma' \oplus At\ i \oplus At\ i{\supset}E$› **and** ‹$S{=}At\ j$› **by** *auto*
then obtain $\Gamma 1$ **where** *eq1*: $\Gamma = \Gamma 1 \oplus At\ i$
and *eq2*: $\Gamma' = \Gamma 1 \oplus At\ i \supset E$
using *midMultiset*[**where** $A{=}At\ i \supset E$ **and** $B{=}At\ i$] **by** *auto*
from *eq2* **and** ‹$\Gamma' \oplus At\ i \oplus E \Rightarrow D \downarrow s$›
have $\Gamma 1 \oplus At\ i \supset E \oplus At\ i \oplus E \Rightarrow D \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus E \oplus At\ i \oplus E \Rightarrow D \downarrow k$
using *inversionImpL0*[**where** $\Gamma{=}\Gamma 1{\oplus}At\ i{\oplus}E$ **and** $B{=}E$ **and** $C{=}D$]
by (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma 1 \oplus E \oplus At\ i \Rightarrow D \downarrow k$ **using** ‹$A{=}S{\supset}T$›
and ‹$S{=}At\ j$› **and** ‹$At\ j \supset T = At\ i \supset E$›
and *IH*[**where** $B{=}E$ **and** $\Gamma{=}\Gamma 1{\oplus}At\ i$] **by** (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma 1 \oplus At\ i \oplus At\ i \supset E \Rightarrow D \downarrow k$
using *provable.ImpL0*[**where** $\Gamma{=}\Gamma 1$ **and** $B{=}E$ **and** $C{=}D$]
by (*auto simp add*:*union-ac*)
then have $\exists\ k.\ \Gamma \oplus S{\supset}T \Rightarrow C \downarrow k$ **using** *eq1* **and** ‹$At\ j \supset T = At\ i \supset E$›
and ‹$S{=}At\ j$› **and** ‹$C{=}D$› **by** *auto*
}
moreover
{**assume** $At\ j \supset T \neq At\ i \supset E$
then have $S{\supset}T \neq At\ i \supset E$ **using** ‹$S{=}At\ j$› **by** *simp*
from ‹$\Gamma \oplus S{\supset}T \oplus S{\supset}T = \Gamma' \oplus At\ i \oplus At\ i \supset E$› **obtain** $\Gamma 2$
where *eq1*: $\Gamma = \Gamma 2 \oplus At\ i \supset E$
and *eq2*: $\Gamma' \oplus At\ i = \Gamma 2 \oplus S{\supset}T \oplus S{\supset}T$
using *midMultiset2*[**where** $A{=}S{\supset}T$ **and** $B{=}At\ i{\supset}E$]
and ‹$S{\supset}T \neq At\ i \supset E$› **by** *auto*
from *eq2* **obtain** $\Gamma 1$ **where** *eq3*: $\Gamma' = \Gamma 1 \oplus S{\supset}T \oplus S{\supset}T$
and *eq4*: $\Gamma 2 = \Gamma 1 \oplus At\ i$
using *midMultiset2*[**where** $A{=}S{\supset}T$ **and** $B{=}At\ i$ **and** $\Gamma{=}\Gamma 2$ **and** $\Gamma'{=}\Gamma'$] **by** *auto*
from *eq4* **and** *eq1* **have** *eq*: $\Gamma = \Gamma 1 \oplus At\ i \oplus At\ i \supset E$ **by** *auto*
from *eq3* **and** ‹$\Gamma' \oplus At\ i \oplus E \Rightarrow D \downarrow s$›
have $\Gamma 1 \oplus S{\supset}T \oplus S{\supset}T \oplus At\ i \oplus E \Rightarrow D \downarrow s$ **by** *simp*
then have $\exists\ k.\ \Gamma 1 \oplus S{\supset}T \oplus At\ i \oplus E \Rightarrow D \downarrow k$
using ‹$A{=}S{\supset}T$› **and** ‹$n{=}s{+}1$›
and *IH*[**where** $\Gamma{=}\Gamma 1{\oplus}At\ i \oplus E$ **and** $B{=}S{\supset}T$ **and** $C{=}D$]

```
        by (auto simp add:union-ac)
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k
        using provable.ImpL0[where Γ=Γ1⊕S⊃T and i=i]
        by (auto simp add:union-ac)
     then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq and ‹C=D› by (auto simp add:union-ac)
     }
   ultimately
   show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by blast
 next
   case (Disj V W)
   then have S⊃T ≠ At i ⊃ E by simp
   from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ At i ⊕ At i ⊃ E› obtain Γ2
       where eq1: Γ = Γ2 ⊕ At i ⊃ E
       and eq2: Γ' ⊕ At i = Γ2 ⊕ S⊃T ⊕ S⊃T
       using midMultiset2[where A=S⊃T and B=At i⊃E] and
       ‹S⊃T ≠ At i ⊃ E› by auto
   from eq2 obtain Γ1 where eq3: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
         and eq4: Γ2 = Γ1 ⊕ At i
        using midMultiset2[where A=S⊃T and B=At i and Γ=Γ2 and Γ'=Γ'] by auto
   from eq4 and eq1 have eq: Γ = Γ1 ⊕ At i ⊕ At i ⊃ E by auto
   from eq3 and ‹Γ' ⊕ At i ⊕ E ⇒ D ↓ s›
       have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ s by simp
   then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ k
       using ‹A=S⊃T› and ‹n=s+1›
       and IH[where Γ=Γ1⊕At i ⊕ E and B=S⊃T and C=D]
       by (auto simp add:union-ac)
   then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k
       using provable.ImpL0[where Γ=Γ1⊕S⊃T and i=i]
       by (auto simp add:union-ac)
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq and ‹C=D› by (auto simp add:union-ac)
 next
   case (Conj V W)
   then have S⊃T ≠ At i ⊃ E by simp
   from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ At i ⊕ At i ⊃ E›
       obtain Γ2 where eq1: Γ = Γ2 ⊕ At i ⊃ E
              and eq2: Γ' ⊕ At i = Γ2 ⊕ S⊃T ⊕ S⊃T
       using midMultiset2[where A=S⊃T and B=At i⊃E] and
       ‹S⊃T ≠ At i ⊃ E› by auto
   from eq2 obtain Γ1 where eq3: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
       and eq4: Γ2 = Γ1 ⊕ At i
       using midMultiset2[where A=S⊃T and B=At i and Γ=Γ2 and Γ'=Γ'] by auto
   from eq4 and eq1 have eq: Γ = Γ1 ⊕ At i ⊕ At i ⊃ E by auto
   from eq3 and ‹Γ' ⊕ At i ⊕ E ⇒ D ↓ s›
```

```
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ k
      using ⟨A=S⊃T⟩ and ⟨n=s+1⟩
      and IH[where Γ=Γ1⊕At i ⊕ E and B=S⊃T and C=D]
      by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k
      using provable.ImpL0[where Γ=Γ1⊕S⊃T and i=i]
      by (auto simp add:union-ac)
   then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq and ⟨C=D⟩ by (auto simp add:union-ac)
 next
    case (Imp V W)
    then have S⊃T ≠ At i ⊃ E by simp
    from ⟨Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ At i ⊕ At i ⊃ E⟩
      obtain Γ2 where eq1: Γ = Γ2 ⊕ At i ⊃ E
      and eq2: Γ' ⊕ At i = Γ2 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=At i⊃E] and
       ⟨S⊃T ≠ At i ⊃ E⟩ by auto
    from eq2 obtain Γ1 where eq3: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      and eq4: Γ2 = Γ1 ⊕ At i
      using midMultiset2[where A=S⊃T and B=At i and Γ=Γ2 and Γ'=Γ']
      by auto
    from eq4 and eq1 have eq: Γ = Γ1 ⊕ At i ⊕ At i ⊃ E by auto
    from eq3 and ⟨Γ' ⊕ At i ⊕ E ⇒ D ↓ s⟩
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ k
      using ⟨A=S⊃T⟩ and ⟨n=s+1⟩
      and IH[where Γ=Γ1⊕At i ⊕ E and B=S⊃T and C=D]
      by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k
      using provable.ImpL0[where Γ=Γ1⊕S⊃T and i=i]
      by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq and ⟨C=D⟩
      by (auto simp add:union-ac)
 next
    case ff
    then have S⊃T ≠ At i ⊃ E by simp
    from ⟨Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ At i ⊕ At i ⊃ E⟩
      obtain Γ2 where eq1: Γ = Γ2 ⊕ At i ⊃ E
      and eq2: Γ' ⊕ At i = Γ2 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=At i⊃E] and
       ⟨S⊃T ≠ At i ⊃ E⟩ by auto
    from eq2 obtain Γ1 where eq3: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      and eq4: Γ2 = Γ1 ⊕ At i
```

```
        using midMultiset2[where A=S⊃T and B=At i and Γ=Γ2 and Γ'=Γ']
        by auto
      from eq4 and eq1 have eq: Γ = Γ1 ⊕ At i ⊕ At i ⊃ E by auto
      from eq3 and ‹Γ' ⊕ At i ⊕ E ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ s by simp
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ E ⇒ D ↓ k
        using ‹A=S⊃T› and ‹n=s+1›
        and IH[where Γ=Γ1⊕At i ⊕ E and B=S⊃T and C=D]
        by (auto simp add:union-ac)
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ At i ⊕ At i ⊃ E ⇒ D ↓ k
        using provable.ImpL0[where Γ=Γ1⊕S⊃T and i=i]
        by (auto simp add:union-ac)
      then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq and ‹C=D›
        by (auto simp add:union-ac)
    qed
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by simp
next
  case (ImpLC Γ' E F G D s)
  then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
  proof (cases S)
    case (Imp V W)
    then have S⊃T ≠ (E∧*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∧*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∧*F)⊃G] and
        ‹S⊃T ≠ (E∧*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ k
        using ‹A=S⊃T› and ‹n=s+1›
        and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S⊃T and C=D]
        by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∧*F)⊃G ⇒ D ↓ k
        using provable.ImpLC[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
        by (auto simp add:union-ac)
   then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
      by (auto simp add:union-ac)
next
    case (Disj V W)
    then have S⊃T ≠ (E∧*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∧*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
```

```
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∧*F)⊃G] and
        ‹S⊃T ≠ (E∧*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ k
        using ‹A=S⊃T› and ‹n=s+1›
        and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S⊃T and C=D]
        by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∧*F)⊃G ⇒ D ↓ k
        using provable.ImpLC[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
        by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
        by (auto simp add:union-ac)
next
    case ff
    then have S⊃T ≠ (E∧*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∧*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∧*F)⊃G] and
        ‹S⊃T ≠ (E∧*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ k
        using ‹A=S⊃T› and ‹n=s+1›
        and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S⊃T and C=D]
        by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∧*F)⊃G ⇒ D ↓ k
        using provable.ImpLC[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
        by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
        by (auto simp add:union-ac)
next
    case (At i)
    then have S⊃T ≠ (E∧*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∧*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∧*F)⊃G] and
        ‹S⊃T ≠ (E∧*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
```

```
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ k
        using ⟨A=S⊃T⟩ and ⟨n=s+1⟩
        and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S⊃T and C=D]
        by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∧*F)⊃G ⇒ D ↓ k
        using provable.ImpLC[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
        by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ⟨C=D⟩
        by (auto simp add:union-ac)
  next
    case (Conj V W)
    have (V∧*W)⊃T = (E∧*F)⊃G ∨ (V∧*W)⊃T ≠ (E∧*F)⊃G by blast
    moreover
      {assume (V∧*W)⊃T = (E∧*F)⊃G
       then have Γ ⊕ (V∧*W)⊃T = Γ'
           using ⟨Γ⊕S⊃T⊕S⊃T = Γ'⊕(E∧*F)⊃G⟩
           and ⟨S=V∧*W⟩ by simp
       then have Γ ⊕ (V∧*W)⊃T ⊕ V⊃(W⊃T) ⇒ D ↓ s
           using ⟨(V∧*W)⊃T = (E∧*F)⊃G⟩
           and ⟨Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s⟩ by auto
       then have ∃ k. Γ ⊕ V⊃(W⊃T) ⊕ V⊃(W⊃T) ⇒ D ↓ k
           using inversionImpLC[where Γ=Γ ⊕ V⊃(W⊃T) and S=V and T=W and B=T
and C=D]
           by (auto simp add:union-ac)
       then have ∃ k. Γ ⊕ V⊃(W⊃T) ⇒ D ↓ k using ⟨S=V∧*W⟩ and ⟨A=S⊃T⟩
           and IH[where B=V⊃(W⊃T) and Γ=Γ] by auto
       then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ⟨C=D⟩ and ⟨S=V∧*W⟩
           and provable.ImpLC by auto
      }
    moreover
      {assume (V∧*W)⊃T ≠ (E∧*F)⊃G
       then have S⊃T ≠ (E∧*F)⊃G using ⟨S=V∧*W⟩ by simp
       from ⟨Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∧*F) ⊃ G⟩ obtain Γ1
           where eq1: Γ = Γ1 ⊕ (E∧*F)⊃G
             and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
           using midMultiset2[where A=S⊃T and B=(E∧*F)⊃G] and
           ⟨S⊃T ≠ (E∧*F)⊃G⟩ by auto
       from eq2 and ⟨Γ' ⊕ E⊃(F⊃G) ⇒ D ↓ s⟩
           have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ s by simp
       then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃(F⊃G) ⇒ D ↓ k
          using ⟨A=S⊃T⟩ and ⟨n=s+1⟩
          and IH[where Γ=Γ1⊕E⊃(F⊃G) and B=S⊃T and C=D]
          by (auto simp add:union-ac)
```

```
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∧*F)⊃G ⇒ D ↓ k
          using provable.ImpLC[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
          by (auto simp add:union-ac)
      then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
          by (auto simp add:union-ac)
      }
    ultimately
    show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by blast
  qed
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by simp
next
  case (ImpLD Γ' E G F D s)
  then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
  proof (cases S)
    case (Imp V W)
    then have S⊃T ≠ (E∨*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∨*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∨*F)⊃G] and
        ‹S⊃T ≠ (E∨*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
        using ‹A=S⊃T› and ‹n=s+1›
        and IH[where Γ=Γ1⊕E⊃G⊕ F⊃G and B=S⊃T and C=D]
        by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∨*F)⊃G ⇒ D ↓ k
        using provable.ImpLD[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
        by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
        by (auto simp add:union-ac)
  next
    case (Conj V W)
    then have S⊃T ≠ (E∨*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∨*F) ⊃ G› obtain Γ1
        where eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
          and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
        using midMultiset2[where A=S⊃T and B=(E∨*F)⊃G] and
        ‹S⊃T ≠ (E∨*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
        have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
```

```
      using ‹A=S⊃T› and ‹n=s+1›
      and IH[where Γ=Γ1⊕E⊃G⊕ F⊃G and B=S⊃T and C=D]
      by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∨*F)⊃G ⇒ D ↓ k
      using provable.ImpLD[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
      by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
      by (auto simp add:union-ac)
  next
    case (At i)
    then have S⊃T ≠ (E∨*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∨*F) ⊃ G› obtain Γ1
      where eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
      and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=(E∨*F)⊃G] and
       ‹S⊃T ≠ (E∨*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
      using ‹A=S⊃T› and ‹n=s+1›
      and IH[where Γ=Γ1⊕E⊃G⊕ F⊃G and B=S⊃T and C=D]
      by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∨*F)⊃G ⇒ D ↓ k
      using provable.ImpLD[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
      by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
      by (auto simp add:union-ac)
  next
    case ff
    then have S⊃T ≠ (E∨*F) ⊃ G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∨*F) ⊃ G› obtain Γ1
      where eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
      and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=(E∨*F)⊃G] and
      ‹S⊃T ≠ (E∨*F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
      using ‹A=S⊃T› and ‹n=s+1›
      and IH[where Γ=Γ1⊕E⊃G⊕ F⊃G and B=S⊃T and C=D]
      by (auto simp add:union-ac)
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∨*F)⊃G ⇒ D ↓ k
      using provable.ImpLD[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
```

```
        by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
        by (auto simp add:union-ac)
  next
    case (Disj V W)
    have (V∨*W)⊃T = (E∨*F)⊃G ∨ (V∨*W)⊃T ≠ (E∨*F)⊃G by blast
    moreover
      {assume (V∨*W)⊃T = (E∨*F)⊃G
       then have Γ ⊕ (V∨*W)⊃T = Γ'
       using ‹S=V∨*W› and ‹Γ⊕S⊃T⊕S⊃T = Γ'⊕ (E∨*F)⊃G› by simp
       then have Γ ⊕ (V∨*W)⊃T ⊕ V⊃T ⊕ W⊃T ⇒ D ↓ s
          using ‹Γ' ⊕E⊃G⊕F⊃G ⇒ D ↓ s› and ‹(V∨*W)⊃T = (E∨*F)⊃G›
          by auto
       then have ∃ k. Γ ⊕ V⊃T ⊕ V⊃T ⊕ W⊃T ⊕ W⊃T ⇒ D ↓ k
          using inversionImpLD[where Γ=Γ⊕V⊃T⊕W⊃T and S=V and T=W and B=T
and C=D]
          by (auto simp add:union-ac)
       then have ∃ k. Γ ⊕ V⊃T ⊕ W⊃T ⊕ W⊃T ⇒ D ↓ k
          using ‹A=S⊃T› and ‹S=V∨*W›
          and IH[where B=V⊃T and Γ=Γ⊕W⊃T⊕W⊃T and C=D]
          by (auto simp add:union-ac)
       then have ∃ k. Γ ⊕ V⊃T ⊕ W⊃T ⇒ D ↓ k using ‹A=S⊃T› and ‹S=V∨*W›
          and IH[where B=W⊃T and Γ=Γ⊕V⊃T and C=D] by auto
       then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using ‹C=D› and
           ‹S=V∨*W› and provable.ImpLD by auto
      }
    moreover
      {assume (V∨*W)⊃T ≠ (E∨*F)⊃G
       then have S⊃T ≠ (E∨*F)⊃G using ‹S=V∨*W› by simp
       from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E∨*F) ⊃ G› obtain Γ1
          where eq1: Γ = Γ1 ⊕ (E∨*F)⊃G
           and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
          using midMultiset2[where A=S⊃T and B=(E∨*F)⊃G] and
          ‹S⊃T ≠ (E∨*F)⊃G› by auto
       from eq2 and ‹Γ' ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s›
          have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ s by simp
       then have ∃ k. Γ1 ⊕ S⊃T ⊕ E⊃G ⊕ F⊃G ⇒ D ↓ k
          using ‹A=S⊃T› and ‹n=s+1›
          and IH[where Γ=Γ1⊕E⊃G⊕ F⊃G and B=S⊃T and C=D]
          by (auto simp add:union-ac)
       then have ∃ k. Γ1 ⊕ S⊃T ⊕ (E∨*F)⊃G ⇒ D ↓ k
          using provable.ImpLD[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
          by (auto simp add:union-ac)
```

```
      then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
        by (auto simp add:union-ac)
    }
  ultimately
  show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by blast
qed
then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by simp
next
case (ImpLL Γ' E F G s D t)
then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
proof (cases S)
  case (At i)
  then have S⊃T ≠ (E⊃F)⊃G by simp
  from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E⊃F)⊃G› obtain Γ1
      where eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=(E⊃F)⊃G] and
      ‹S⊃T ≠ (E⊃F)⊃G› by auto
  from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
  then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕E⊕ F⊃G and B=S⊃T and C=F and m=s]
      by (auto simp add:union-ac)
  moreover
  from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ G ⇒ D ↓ t by simp
  then have ∃ k. Γ1 ⊕ S⊃T ⊕ G ⇒ D ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕G and B=S⊃T and C=D and m=t]
      by (auto simp add:union-ac)
  ultimately
  have ∃ k. Γ1 ⊕ S⊃T ⊕ (E⊃F)⊃G ⇒ D ↓ k
      using provable.ImpLL[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
      by (auto simp add:union-ac)
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
      by (auto simp add:union-ac)
next
  case (Conj V W)
  then have S⊃T ≠ (E⊃F)⊃G by simp
  from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E⊃F)⊃G› obtain Γ1
      where eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
```

```
      using midMultiset2[where A=S⊃T and B=(E⊃F)⊃G] and
      ‹S⊃T ≠ (E⊃F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕E⊕ F⊃G and B=S⊃T and C=F and m=s]
      by (auto simp add:union-ac)
    moreover
    from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ G ⇒ D ↓ t by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ G ⇒ D ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕G and B=S⊃T and C=D and m=t]
      by (auto simp add:union-ac)
    ultimately
    have ∃ k. Γ1 ⊕ S⊃T ⊕ (E⊃F)⊃G ⇒ D ↓ k
      using provable.ImpLL[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
      by (auto simp add:union-ac)
    then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
      by (auto simp add:union-ac)
  next
    case (Disj V W)
    then have S⊃T ≠ (E⊃F)⊃G by simp
    from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E⊃F)⊃G› obtain Γ1
      where eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
        and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
      using midMultiset2[where A=S⊃T and B=(E⊃F)⊃G] and
      ‹S⊃T ≠ (E⊃F)⊃G› by auto
    from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕E⊕ F⊃G and B=S⊃T and C=F and m=s]
      by (auto simp add:union-ac)
    moreover
    from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t›
      have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ G ⇒ D ↓ t by simp
    then have ∃ k. Γ1 ⊕ S⊃T ⊕ G ⇒ D ↓ k
      using ‹A=S⊃T› and ‹n=s+t+1›
      and IH[where Γ=Γ1⊕G and B=S⊃T and C=D and m=t]
      by (auto simp add:union-ac)
    ultimately
```

```
  have ∃ k. Γ1 ⊕ S⊃T ⊕ (E⊃F)⊃G ⇒ D ↓ k
    using provable.ImpLL[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
    by (auto simp add:union-ac)
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
    by (auto simp add:union-ac)
next
  case ff
  then have S⊃T ≠ (E⊃F)⊃G by simp
  from ‹Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E⊃F)⊃G› obtain Γ1
    where eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
    and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
    using midMultiset2[where A=S⊃T and B=(E⊃F)⊃G] and
    ‹S⊃T ≠ (E⊃F)⊃G› by auto
  from eq2 and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s›
    have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
  then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ k
    using ‹A=S⊃T› and ‹n=s+t+1›
    and IH[where Γ=Γ1⊕E⊕ F⊃G and B=S⊃T and C=F and m=s]
    by (auto simp add:union-ac)
  moreover
  from eq2 and ‹Γ' ⊕ G ⇒ D ↓ t›
    have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ G ⇒ D ↓ t by simp
  then have ∃ k. Γ1 ⊕ S⊃T ⊕ G ⇒ D ↓ k
    using ‹A=S⊃T› and ‹n=s+t+1›
    and IH[where Γ=Γ1⊕G and B=S⊃T and C=D and m=t]
    by (auto simp add:union-ac)
  ultimately
  have ∃ k. Γ1 ⊕ S⊃T ⊕ (E⊃F)⊃G ⇒ D ↓ k
    using provable.ImpLL[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
    by (auto simp add:union-ac)
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k using eq1 and ‹C=D›
    by (auto simp add:union-ac)
next
  case (Imp V W)
  have (V⊃W)⊃T = (E⊃F)⊃G ∨ (V⊃W)⊃T ≠ (E⊃F)⊃G by blast
  moreover
    {assume (V⊃W)⊃T = (E⊃F)⊃G
     then have Γ ⊕ (V⊃W)⊃T = Γ' using ‹S=V⊃W›
       and ‹Γ ⊕S⊃T⊕S⊃T = Γ' ⊕ (E⊃F)⊃G› by simp
     then have Γ ⊕ (V⊃W)⊃T ⊕ V ⊕ W⊃T ⇒ W ↓ s
       using ‹(V⊃W)⊃T = (E⊃F)⊃G›
       and ‹Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s› by auto
     then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⊕ W⊃T ⊕ W⊃T ⇒ W ↓ k
```

```
            using twoDB[where Γ=Γ⊕V⊕W⊃T and C=V and D=W and B=T and E=W
and n=s]
            by (auto simp add:union-ac)
        then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⊕ W⊃T ⇒ W ↓ k
            using ⟨A=S⊃T⟩ and ⟨S=V⊃W⟩
            and IH[where B=W⊃T and Γ=Γ⊕ V ⊕ V ⊕ W⊃T and C=W]
            by auto
        then have ∃ k. Γ ⊕ V ⊕ V ⊕ W⊃T ⇒ W ↓ k
            using ⟨A=S⊃T⟩ and ⟨S=V⊃W⟩
            and IH[where B=W⊃T and Γ=Γ⊕ V ⊕ V and C=W] by auto
        then have ∃ k. Γ ⊕ V ⊕ W⊃T ⇒ W ↓ k
            using ⟨A=S⊃T⟩ and ⟨S=V⊃W⟩
            and IH[where B=V and Γ=Γ⊕ W⊃T and C=W]
            by (auto simp add:union-ac)
        moreover
        from ⟨Γ ⊕ (V⊃W)⊃T = Γ'⟩ and ⟨Γ' ⊕ G ⇒ D ↓ t⟩
            have Γ⊕(V⊃W)⊃T ⊕ T ⇒ D ↓ t
            using ⟨(V⊃W)⊃T = (E⊃F)⊃G⟩
            by auto
        then have ∃ k. Γ ⊕ T ⊕ T ⇒ D ↓ k
            using inversionImpLL[where Γ=Γ⊕T and S=V and T=W and B=T and C=D]
            by (auto simp add:union-ac)
        then have ∃ k. Γ ⊕ T ⇒ D ↓ k using ⟨A=S⊃T⟩ and IH by auto
        ultimately
            have ∃ k. Γ ⊕ (V⊃W)⊃T ⇒ D ↓ k
            using provable.ImpLL by auto
        then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
            using ⟨S=V⊃W⟩ and ⟨C=D⟩ by simp
       }
      moreover
       {assume (V⊃W)⊃T ≠ (E⊃F)⊃G
        then have S⊃T ≠ (E⊃F)⊃G using ⟨S=V⊃W⟩ by simp
        from ⟨Γ ⊕S⊃T ⊕ S⊃T = Γ' ⊕ (E⊃F)⊃G⟩ obtain Γ1
            where eq1: Γ = Γ1 ⊕ (E⊃F)⊃G
            and eq2: Γ' = Γ1 ⊕ S⊃T ⊕ S⊃T
            using midMultiset2[where A=S⊃T and B=(E⊃F)⊃G] and
            ⟨S⊃T ≠ (E⊃F)⊃G⟩ by auto
        from eq2 and ⟨Γ' ⊕ E ⊕ F⊃G ⇒ F ↓ s⟩
            have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ s by simp
        then have ∃ k. Γ1 ⊕ S⊃T ⊕ E ⊕ F⊃G ⇒ F ↓ k
            using ⟨A=S⊃T⟩ and ⟨n=s+t+1⟩
            and IH[where Γ=Γ1⊕E⊕ F⊃G and B=S⊃T and C=F and m=s]
            by (auto simp add:union-ac)
```

```
      moreover
      from eq2 and ‹Γ′ ⊕ G ⇒ D ↓ t›
          have Γ1 ⊕ S⊃T ⊕ S⊃T ⊕ G ⇒ D ↓ t by simp
      then have ∃ k. Γ1 ⊕ S⊃T ⊕ G ⇒ D ↓ k
          using ‹A=S⊃T› and ‹n=s+t+1›
          and IH[where Γ=Γ1⊕G and B=S⊃T and C=D and m=t]
          by (auto simp add:union-ac)
      ultimately
      have ∃ k. Γ1 ⊕ S⊃T ⊕ (E⊃F)⊃G ⇒ D ↓ k
          using provable.ImpLL[where Γ=Γ1⊕S⊃T and A=E and B=F and C=G]
          by (auto simp add:union-ac)
      then have ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k
          using eq1 and ‹C=D› by (auto simp add:union-ac)
     }
    ultimately show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by blast
    qed
  then show ∃ k. Γ ⊕ S⊃T ⇒ C ↓ k by simp
  qed
 then show ∃ k. Γ ⊕ A ⇒ C ↓ k using ‹A=S⊃T› by simp
 qed
qed

lemma G3iImplication:
  assumes Γ ⊕ A⊃B ⇒ A ↓ n and Γ ⊕ B ⇒ E ↓ m
  shows ∃ k. Γ ⊕ A⊃B ⇒ E ↓ k
proof−
  from ‹Γ ⊕ B ⇒ E ↓ m› have Γ ⊕ A⊃B ⊕ B ⇒ E ↓ m using dpWeak[where Γ=Γ⊕B and
A=A⊃B] by (auto simp add:union-ac)
  then have ∃ k. Γ ⊕ A⊃B ⊕ A⊃B ⇒ E ↓ k using ‹Γ ⊕ A⊃B ⇒ A ↓ n› and ImpLClassi-
cal[where Γ=Γ⊕A⊃B and D=A and B=B and S=E]
      by (auto simp add:union-ac)
  then show ∃ k. Γ ⊕ A⊃B ⇒ E ↓ k using contract by auto
qed

end
```