Generic Methods for Formalising Sequent Calculi Applied to Provability Logic

Jeremy E. Dawson and Rajeev Goré

Logic and Computation Group, School of Computer Science
The Australian National University, Canberra ACT 0200, Australia
`http://users.rsise.anu.edu.au/~jeremy/`
`http://users.rsise.anu.edu.au/~rpg/`

Abstract. We describe general methods for reasoning about multiset-based sequent calculi and show how they allow us to mix-and-match shallow and deep embeddings. Our methods are modular, permit explicit structural rules, and are widely applicable to other sequent systems, even to other derivation systems. We also describe new axiomatic type classes relevant to multisets and sequents of multisets. We demonstrate the benefits of our approach by formalising a variant of a recent, non-trivial, pen-and-paper proof of cut-admissibility for the provability logic **GL** in Isabelle/HOL. Our work confirms that the controversy surrounding the proof of cut-admissibility for **GL** has finally been settled.

1 Introduction

Sequent calculi provide a rigorous basis for meta-theoretic studies of various logics. The central theorem is cut-elimination/admissibility, which states that detours through lemmata can be avoided, since it can help to show many important logical properties like consistency, interpolation, and Beth definability. Cut-free sequent calculi are also used for automated deduction, for nonclassical extensions of logic programming, and for studying the connection between normalising lambda calculi and functional programming. Sequent calculi, and their extensions, therefore play an important role in logic and computation.

Meta-theoretic reasoning about sequent calculi is error-prone because it involves checking many combinatorial cases, some being very difficult, but many being similar. Invariably, authors resort to expressions like "the other cases are similar", or "we omit details". The literature contains many examples of meta-theoretic proofs containing serious and subtle errors in the original pencil-and-paper proofs. For example, the cut-elimination theorem for the modal "provability logic" **GL**, where $\Box\varphi$ can be read as "φ is provable in Peano Arithmetic", has a long and chequered history which has only recently been resolved [5].

Various proof assistants and logical frameworks now allow us to formalise basic reasoning about sequent calculi [8, 9, 3], and some even allow for automatic cut-elimination. But in all of these, it is not possible to add and reason about arbitrary structural rules since they invariably use a shallow embedding that identifies the properties of the object logic with those of some fixed meta-logic.

Here, we describe general methods for formalising the proof-theory of traditional, propositional, multiset-based sequent calculi containing explicit structural rules in Isabelle/HOL. We then show how we used them to formalise a proof of cut-admissibility for **GL** based on, but different from, that of [5].

In Section 2, we briefly describe traditional sequent calculi, discuss the need for multisets, and describe the controversy surrounding the cut-elimination theorem for the set-based sequent system GLS for provability logic **GL** . In Section 3 we describe techniques and functions we have defined for reasoning about derivations, and show how we formalise formulae, sequents and rules, showing some of the **GL** sequent rules as examples. In Section 4 we describe an Isabelle axiomatic type class which we developed to facilitate reasoning about multisets of formulae, and sequents based on them. This development explores the interaction between lattice ($\wedge$, $\vee$, $\leq$) and "arithmetic" ($+$, $-$, 0, $\leq$) operations. In Section 5 we discuss those aspects of our Isabelle proofs which we tried to make as general as possible, and how they are useful for proving cut-elimination and other results in arbitrary sequent calculi which meet the associated preconditions. In Section 6 we describe the cut-admissibility proof for GLS.

We assume the reader is familiar with basic proof-theory, ML and Isabelle/HOL. Our Isabelle code can be found at `http://users.rsise.anu.edu.au/~jeremy/isabelle/200x/seqms/`. Some of this work was reported informally in [4].

2 Sequents, Multisets, Sets and Provability Logic

Proof-theorists typically work with sequents $\Gamma \vdash \Delta$ where Γ and Δ are "collections" of formulae. The "collections" found in the literature increase in complexity from simple sets for classical logic, to multisets for linear logic, to ordered lists for substructural logics, to complex tree structures for display logics. A sequent rule typically has a rule name, a (finite) number of premises, a side-condition and a conclusion. Rules are read top-down as "if all the premises hold then the conclusion holds". A derivation of the judgement $\Gamma \vdash \Delta$ is typically a finite tree of judgements with root $\Gamma \vdash \Delta$ where parents are obtained from children by "applying a rule". We use "derivation" to refer to a proof *within* a calculus, reserving "proof" for a meta-theoretic proof of a theorem *about* the calculus.

Provability logic **GL** is obtained by adding Löb's axiom $\Box(\Box A \to A) \to \Box A$ to propositional normal modal logic K. Its Kripke semantics is based on rooted transitive Kripke frames without infinite forward chains. It rose to prominence when Solovay showed that $\Box A$ could be interpreted as "A is provable in Peano Arithmetic" [10]. An initial proof-theoretic account was given by Leivant in 1981 when he "proved" cut-elimination for a set-based sequent calculus for **GL** [6]. But Valentini in 1983 found a simple counter-example and gave a new cut-elimination proof [11]. The issue seemed to have been settled, but in 2001, Moen [7] claimed that Valentini's transformations don't terminate if the sequents $\Gamma \vdash \Delta$ are based on multisets. There is of course no *a priori* reason why a proof based on sets should not carry over with some modification to a proof based on multisets. The issue was recently resolved by Goré and Ramanayake [5] who showed that Moen is incorrect, and that Valentini's proof using multisets is *mostly* okay. Many such examples exist in the literature.

The sequent system GLS for the logic **GL** as given by Goré and Ramanayake in [5], like Valentini's, contains explicit weakening and contraction rules and, modulo these, a typical (additive) set of rules for the classical logical connectives $\neg, \wedge, \vee, \rightarrow$. The one additional rule GLR which characterises **GL** is shown below:

$$\frac{\Box X, X, \Box B \vdash B}{\Box X \vdash \Box B}\ GLR \text{ or } \quad GLR(\mathrm{B}) \quad \text{or} \quad GLR(\mathrm{X,B})$$

The rule is unusual since the principal formula $\Box B$ changes polarity from conclusion to premise. To identify the principal formula involved, or all the formulae, we may refer to it as $GLR(B)$, or $GLR(X, B)$. Note that our formalisation does not regard the cut (or multicut) rules shown below as being part of the system:

$$(\text{cut})\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta} \qquad (\text{multicut})\ \frac{\Gamma \vdash A^n, \Delta \qquad \Gamma, A^m \vdash \Delta}{\Gamma \vdash \Delta}$$

Thus our results will be lemmata of the form: if $\Gamma \vdash A, \Delta$ is (cut-free) derivable and $\Gamma, A \vdash \Delta$ is (cut-free) derivable then $\Gamma \vdash \Delta$ is (cut-free) derivable.

3 Reasoning About Derivations and Derivability

In previous work [1], we describe our initial attempts to formalise display calculi in various logical frameworks and describe why we selected Isabelle/HOL. When we model a derivation tree, explicitly, we do so using the capability in HOL of defining types like the datatypes of SML, see §3.1.

3.1 Deep and Shallow Embeddings

In [2] we proved cut admissibility for $\boldsymbol{\delta}$**RA**, the display calculus for relation algebras. The proof was based on a proof of Belnap, which applied generally to display calculi whose inference rules satisfied certain properties. In that paper we described the formalisation as a "deep embedding". We now believe that to describe a particular formalisation as either a "deep embedding" or a "shallow embedding" over-simplifies the issue as we now discuss.

In [2], we modelled a derivation tree in HOL as a recursive structure which consists of a root sequent (which should be the conclusion of some rule), and an associated list of (sub)trees (each of which should derive a premise of that rule). The type of explicit derivation trees is a recursive datatype, such as:

```
datatype 'a dertree = Der 'a ('a dertree list)
     | Unf 'a (* unfinished leaf which remains unproved *)
```

We then had to define a predicate on such a tree, saying whether it was in fact correctly based on the given rules, and so represented a valid derivation. We modelled a sequent rule as an object in HOL consisting of a list of premises and a conclusion, each of which was a sequent in the language of the logic (of relation algebras). Notably, our language of formulae included "variables" which could be instantiated with formulae, so we defined functions for such substitutions. Thus it is more accurate to say in [2], we used a deep embedding for *derivations*, *rules* and *variables*, since we modelled each of these features of the proof-theory explicitly rather than identifying them with the same features of Isabelle.

In alternative approaches to work of this type, the set of derivable sequents can be modelled as an inductively defined set `ders` (say), where the derivation tree does not exist, but it "happens" in the course of proving that a specific sequent is derivable. That is, derivability in the sequent calculus studied equates to derivability in Isabelle, giving a shallow embedding of *derivations*.

In such alternative approaches, the rules of the sequent calculus can either be encoded explicitly or they can be encoded in the clauses for the inductive definition of `ders`. In the latter case, the rules of the sequent calculus under study are encoded as theorems of Isabelle. Thus, even with a shallow embedding of derivations, we have a choice of either a deep or a shallow embedding of *rules*:

Deep Rules and Shallow Derivation:

$(\{\Gamma \vdash \mathtt{P},\ \ \Gamma \vdash \mathtt{Q}\},\ \ \Gamma \vdash \mathtt{P} \wedge \mathtt{Q}) \in \mathtt{rules}$

$[|\ (\mathtt{ps}, \mathtt{c}) \in \mathtt{rules}\ ;\ \mathtt{ps} \subseteq \mathtt{ders}\ |] ==> \mathtt{c} \in \mathtt{ders}$

Shallow Rules and Shallow Derivations:

$[|\ \Gamma \vdash \mathtt{P} \in \mathtt{ders}\ ;\ \Gamma \vdash \mathtt{Q} \in \mathtt{ders}\ |] ==> \Gamma \vdash \mathtt{P} \wedge \mathtt{Q} \in \mathtt{ders}.$

Finally, compared to [2], it is much more common to let the variables in a rule be the variables of the theorem prover, and for the theorem prover to do the substitution. In general, however, a deep embedding of one feature can be combined with a shallow embedding of another.

Our experience is that shallow embeddings generally permit easier proofs, but sometimes they are inadequate to express a desired concept. For example, with a deep embedding of rules it is easy to express that one set of rules is a subset of another set. With a deep embedding of derivation trees it is easy to express that one property of derivation trees is the conjunction of two other properties, or that a derivation tree has a particular height, whereas to express such things using `derl`, `derrec`, etc (see §3.2), one would have to redefine these predicates incorporating the particular properties of interest. Indeed, in this work (see §6) we discuss how we found a shallow embedding of derivability inadequate.

We have developed the code on which [2] was based to reflect the fact that Belnap's original proof applied generally to display calculi satisfying certain conditions. Some of these conditions required a deep embedding of variables (and, therefore, of the rules), for example, that no structure variable appears twice in the conclusion of the statement of a rule. For others, a shallow embedding of variables was adequate, since the condition applied equally to the statement of the rule and to all its substitution instances, for example that a premise of a rule contains no variable not contained in the conclusion.

3.2 Our General Derivability Predicates

We now briefly describe the functions we used to describe derivability. A fuller account is given in [4]. This framework is general in that a rule merely consists of "premises" and a "conclusion", and is independent of whether the things derived are formulae, sequents, or other constructs, but we will refer to them as sequents.

```
types 'a psc = "'a list * 'a"       (* single step inference *)
consts
  derl, adm :: "'a psc set => 'a psc set"
  dersl     :: "'a psc set => ('a list * 'a list) set"
  dercsl    :: "'a psc set => ('a list list * 'a list) set"
  derrec    :: "'a psc set => 'a set => 'a set"
  dersrec   :: "'a psc set => 'a set => 'a list set"
```

An inference rule `'a psc` is a list of premises `ps` and a conclusion `c`. Then, `derl rls` is the set of rules derivable from the rule set `rls`, and `derrec rls prems` is the set of sequents derivable using rules `rls` from the set `prems` of premises. These were defined as inductive sets, using also the functions `dersl` and `dersrec`. So $(\mathtt{ps}, \mathtt{c}) \in$ `derl rls` reflects the shape of a derivation tree: `ps` is a list of exactly the premises used, in the correct order, whereas `c` $\in$ `derrec rls prems` holds even if the set of premises `prems` contains extra sequents.

The auxiliary function `dersrec` represents several sequents, all derivable from the premises. The auxiliary function `dersl` represents several derivation trees side by side; $(\mathtt{ps}, \mathtt{cs}) \in$ `dersl rls` when `ps` is the concatenation of their lists of premises, and `cs` is the list of their conclusions. The function `dercsl` is similar, but it shows which premises are part of which tree. That is, $(\mathtt{pss}, \mathtt{cs}) \in$ `dercsl rls` implies $(\mathtt{concat\ pss}, \mathtt{cs}) \in$ `dersl rls`. Curiously, we have used this framework for several years, and only recently we found the need to redefine `derl` using `dercsl` rather than `dersl` as the auxiliary function.

The key clauses for these functions are shown below:

```
  dtderI : "[| (ps, concl) : pscrel ; (pss, ps) : dersl pscrel |]
              ==> (pss, concl) : derl pscrel"
  derI   : "[| (ps, concl) : pscrel ; ps : dersrec pscrel prems |]
              ==>  concl : derrec pscrel prems"
```

We obtained the expected results linking `derl` and `derrec`, and a number of results expressing transitivity of derivation and the results of derivation using derived rules, for example:

```
derl_trans : "[| (?ps, ?c) : derl ?rls; (?x, ?ps) : dersl ?rls |]
                 ==> (?x, ?c) : derl ?rls"
derl_deriv_eq : "derl (derl ?rls) = derl ?rls"
derrec_trans_eq : "derrec ?rls (derrec ?rls ?prems) = derrec ?rls ?prems"
```

A simplified verson of the induction principle generated by the definition of the inductive set `derrec` is as follows:

$$\frac{x \in \mathit{derrec\ rls\ prems} \quad \forall c \in \mathit{prems}.P\ c \quad \forall (ps, c) \in \mathit{rls}.(\forall p\ \mathit{in}\ ps.P\ p) \Rightarrow P\ c}{P\ x}$$

The principle says that our rules must preserve the property P from premises to conclusions. We contrast this principle with induction on the height of derivations where it is possible, and sometimes necessary, to transform a sub-tree in some height-preserving (or height-reducing) way, and use the assumption that the transformed tree has property P. Such transformations are not available when using the induction principle above, and were not needed in our earlier work.

Where we have a property of two derivations, such as cut-admissibility, we need a more complex induction principle which we derived, where the left tree derives cl and the right tree derives cr:

$$\frac{\begin{array}{c} cl \in derrec\ rlsl\ \{\} \quad cr \in derrec\ rlsr\ \{\} \\ \forall (lps, lc) \in rlsl, (rps, rc) \in rlsr.\ (\forall lp\ in\ lps. P\ lp\ rc) \wedge (\forall rp\ in\ rps. P\ lc\ rp) \Rightarrow P\ lc\ rc \end{array}}{P\ cl\ cr}$$

Finally, (ps, c) is an admissible rule iff: if all premises in ps are derivable, then c is too: $(ps, c) \in$ `adm rls` $\iff$ $(set\ ps \subseteq$ `derrec rls` $\Rightarrow$ $c \in$ `derrec rls`$)$.

We obtained the following four results, which were surprisingly tricky because `adm` is not monotonic:

```
"derl ?rls <= adm ?rls"              "adm (adm ?rls) = adm ?rls"
"adm (derl ?rls) = adm ?rls"         "derl (adm ?rls) = adm ?rls"
```

3.3 Sequents, Formulae and the GLS rules

We define a language of formula connectives, formula variables and primitive (atomic) propositions:

```
datatype formula = FC string (formula list) (* formula connective *)
                 | FV string                  (* formula variable *)
                 | PP string              (* primitive proposition *)
```

The formula connectives are fixed for each logic, but we make the datatype as general as possible, replacing connectives such as $\wedge$, $\vee$, etc, with `FC` *name*:

```
consts Btimes :: "formula => formula => formula" ("_ && _" [67,68] 67)
       Bplus  :: "formula => formula => formula" ("_ v _" [63,64] 63)
defs   Btimes "Btimes f g == FC ''Btimes'' [f, g]"
       Bplus  "Bplus f g == FC ''Bplus'' [f, g]"
```

This permits a single definition of the immediate (proper) subformula relation, `ipsubfml`, which will not need to be changed when new connectives are added:

```
consts ipsubfml :: "(formula * formula) set"
inductive "ipsubfml" (* proper immediate subformula relation *)
  intros  ipsI : "P : set Ps ==> (P, FC conn Ps) : ipsubfml"
```

A sequent is a pair of multisets of formulae, written $\Gamma \vdash \Delta$ or $\Gamma \Rightarrow \Delta$.

```
datatype 'a sequent =
 Sequent "'a multiset" "'a multiset"    ("((_)/ |- (_))" [6,6] 5)
```

Given a rule such as $(\vdash \wedge)$ in the form on the left,

$$\mathcal{C}_s = \frac{\vdash A \quad \vdash B}{\vdash A \wedge B} \qquad \mathcal{C}_e = \frac{X \vdash Y, A \quad X \vdash Y, B}{X \vdash Y, A \wedge B}$$

we define functions `extend` and `pscmap`, such that (for example)

$$\texttt{extend}\ (X \vdash Y)\ (U \vdash V) = (X + U) \vdash (Y + V)$$
$$\mathcal{C}_e = \texttt{pscmap}\ (\texttt{extend}\ (X \vdash Y))\ \mathcal{C}_s$$

Then `extrs` S means the set of all extensions of all rules in the set S.

```
consts extend :: "'a sequent => 'a sequent => 'a sequent"
       extrs  :: "'a sequent psc set => 'a sequent psc set"
defs   extend_def : "extend fmls seq == seq + fmls"
       pscmap_def : "pscmap f (ps, c) = (map f ps, f c)"
```

Then we define `glss`, the set of rules of GLS; we show just a selection. Sets `wkrls` and `ctrrls` A refer to weakening and contraction (on A) rules.

```
inductive "glir" (* right introduction rules *)
intros
 andr : "([{#} |- {#A#}, {#} |- {#B#}], {#} |- {#A && B#}) : glir"
 orr1 : "([{#} |- {#A#}], {#} |- {#A v B#}) : glir"

inductive "glne" (* rules before being extended *)
intros wkI  : "(ps, c) : wkrls ==> (ps, c) : glne"
       ctrI : "(ps, c) : ctrrls A ==> (ps, c) : glne"
       ilI  : "(ps, c) : glil ==> (ps, c) : glne"
       irI  : "(ps, c) : glir ==> (ps, c) : glne"

inductive "glr B"
intros I : "([mset_map Box X + X + {#Box B#} |- {#B#}],
             mset_map Box X |- {#Box B#}) : glr B"

inductive "glss"
intros axiom : "([], {#A#} |- {#A#}) : glss"
        extI : "psc : glne ==> pscmap (extend flr) psc : glss"
        glrI : "psc : glr B ==> psc : glss"
```

Note that X is a multiset, and $\Box X$ is informal notation for applying $\Box$ to each member of X; this is formalised as `mset_map`, used in the formalised *GLR* rule. Using a similar notation we write $\Box B^k$ for $(\Box B)^k$, the multiset containing n copies of $\Box B$. Development of `mset_map` and relevant lemmas is in the source files `Multiset_no_le.{thy,ML}`. These results also show multisets form a monad.

4 An Axiomatic Type Class for Multisets and Sequents

Isabelle provides a theory of finite multisets with an ordering based on an underlying ordering on members: given multiset M, we obtain a multiset N less than M by removing an occurrence of some m from M, and replacing it with any number of occurrences of elements smaller than m. We defined a different ordering $\leq$ analogous to $\subseteq$ for sets: $N \leq M$ if, for all x, N contains no more occurrences of x than does M.

An axiomatic type class is characterised by a number of axioms, which hold for all members of a type in the type class. The axioms can be used to prove theorems, which then apply and can be used efficiently for any type which is in the class. The multiset operators $\leq$, $+$, $-$ and 0 have several useful properties, which are described by the following axiomatic type classes `pm0` and `pm_ge0`:

```
axclass pm0 < comm_monoid_add, minus
  pm0_plus_minus  : "A + B - A = B"
  pm0_minus_minus : "A - B - C = A - (B + C)"
```

Thus for any type in class `pm0`, the operations $+$ and 0 form a commutative monoid, the type allows the $-$ operator, and the two axioms given (above) hold.

```
axclass pm_ge0 < pm0, ord, leq
  all_ge0        : "0 <= A"
  diff_0_le      : "(m - n = 0) = (m <= n)"
  le_add_diff_eq : "B <= A ==> B + (A - B) = A"
  pm_ge0_less_le : "(x < y) = (x <= y & x ~= y)"
  pg_leq_def     : "a [= b == a <= b"
```

Class `pm_ge0` also has an $\leq$ operator, 0 is the smallest element via `all_ge0`, and `le_add_diff_eq` also holds. For a new type, the other axioms could be made definitions, but for natural numbers, $\leq$ and $<$ are already defined. The naturals obey all these axioms. An explanation of `[=` is below.

Lemma 1. *Multisets are in `pm0` and `pm_ge0` using our definition of $\leq$, and, if Γ and Δ are of any type in the classes `pm0` or `pm_ge0`, then so is sequent $\Gamma \vdash \Delta$.*

Isabelle contains "simplification procedures", which will simplify $a-b+c+b$ to $a+c$ for integers, and which will simplify $a+b+c-b$ to $a+c$ for integers *or* naturals. We have been able to apply almost all of the ones which apply to naturals, to types of the classes `pm0` and `pm_ge0`.

It is easy to show that, in the obvious way, multisets form a lattice. Isabelle's lattice type class is based on using `[=` as the order operator. By including the definitional axiom `pg_leq_def` above, we were able to show:

Lemma 2. *Any type of class `pm_ge0` forms a lattice.*

Proof. The definitions of meet and join are shown below:

```
"pg_meet ?c ?d == ?c - (?c - ?d)"  "pg_join ?c ?d == ?c + (?d - ?c)"
```

From these it is possible to prove the axioms for a lattice at some length, and so any type in the class `pm_ge0` is also in Isabelle's class `lattice`. $\dashv$

We produced a number of equivalent definitions characterising atoms of the lattice, and properties of atomic elements (our definition of `atomic` includes 0). Thus we showed that "x is atomic" is equivalent to each of the following:

$$\forall y < x.\ y = 0$$

$$\forall z.\ x \leq z \vee (pg\_meet\ x\ z = 0) \qquad \forall ab.\ x \leq a + b \Rightarrow x \leq a \vee x \leq b$$

$$\forall ab.\ 0 < a \wedge 0 < b \Rightarrow a + b \neq x \qquad \forall ab.\ a < x \wedge b < x \Rightarrow a + b \neq x$$

The source files for this development are `pmg*.{thy,ML}`.

5 Generalising Cut-Admissibility Proofs

We now show how our previous work on multicut admissibility for LK can be formulated to make it as general as possible. In §3 we mentioned the induction principle used for deriving cut-admissibility, or indeed any property P of two-premise subtrees. In the diagram below, to prove $P(\mathcal{C}_l, \mathcal{C}_r)$, the induction assumption is that $P(\mathcal{P}_{li}, \mathcal{C}_r)$ and $P(\mathcal{C}_l, \mathcal{P}_{rj})$ hold for all i and j:

$$\dfrac{\dfrac{\mathcal{P}_{l1} \ldots \mathcal{P}_{ln}}{\mathcal{C}_l}\,\rho_l \qquad \dfrac{\mathcal{P}_{r1} \ldots \mathcal{P}_{rm}}{\mathcal{C}_r}\,\rho_r}{?}\,(cut\ ?)$$

A proof of $P(\mathcal{C}_l, \mathcal{C}_r)$ using this induction assumption inevitably proceeds by cases on the actual rules ρ_l and ρ_r, and for a cut-formula A, on whether it is principal in either or both of ρ_l and ρ_r. But we also use induction on the size of the cut-formula, or, more generally, on some well-founded relation on formulae. So we actually consider a property $P\ A\ (cl, cr)$ where psl are the premises $\mathcal{P}_{l1} \ldots \mathcal{P}_{ln}$ of rule ρ_l, and cl is its conclusion $\mathcal{C}_l$, and analogously for ρ_r, and A is the cut-formula. In proving $P\ A\ (\mathcal{C}_l, \mathcal{C}_r)$, in addition to the inductive assumption above, we assume that $P\ A'\ (da, db)$ holds generally for A' smaller than A and all "`rls`-derivable" sequents da and db, ie, da and db derivable from the empty set of premises using rules from rls. These intuitions give the following definition `gen_step2ssr` of a condition which permits one step of the inductive proof.

Definition 1 (`gen_step2ssr`). *For a formula* `A`*, a property* `P`*, a subformula relation* `sub`*, a set of rules* `rls`*, inference rule instances* `(psl, cl)`*, and* `(psr, cr)`*, the property* `gen_step2ssr` *means that:*

if all the following hold:

- *for all subformulae* `A'` *of* `A` *and all* `rls`*-derivable sequents* `da` *and* `db`*,* `P A' (da, db)` *holds*
- *for each* `pa` *in* `psl`*,* `P A (pa, cr)` *holds;*
- *for each* `pb` *in* `psr`*,* `P A (cl, pb)` *holds;*

then `P A (cl, cr)` *holds*

```
gen_step2ssr_simp :
"gen_step2ssr ?P ?A ?sub ?rls ((?psl, ?cl), (?psr, ?cr)) =
( (ALL A'. (A', ?A) : ?sub -->
   (ALL da:derrec ?rls {}. ALL db:derrec ?rls {}. ?P A' (da, db))) -->
  (ALL pa:set ?psl. ?P ?A (pa, ?cr)) -->
  (ALL pb:set ?psr. ?P ?A (?cl, pb)) --> ?P ?A (?cl, ?cr) )"
```

The main theorem `gen_step2ssr_lem` below for proving an arbitrary property P states that if the step of the inductive proof is possible for all cases of final rule instances (psl, cl) and (psr, cr) on each side, then P holds in all cases.

Theorem 1. *If `A` is in the well-founded part of the subformula relation `sub`; sequents `seql` and `seqr` are `rls`-derivable ; and for all formulae `A`, and all rules `(psl, cl)` and `(psr, cr)`, our induction step condition `gen_step2ssr P A sub rls ((psl, cl), (psr, cr))` holds, then `P A (seql, seqr)` holds.*

```
gen_step2ssr_lem :
 "[| ?A : wfp ?sub ;
     ?seql : derrec ?rls {} ; ?seqr : derrec ?rls {} ;
     ALL A. ALL (psl, cl):?rls. ALL (psr, cr):?rls.
         gen_step2ssr ?P A ?sub rls ((psl, cl), (psr, cr)) |]
     ==> ?P ?A (?seql, ?seqr)"
```

This enables us to split up an inductive proof, by showing, separately, that `gen_step2ssr` holds for particular cases of the final rules (psl, cl) and (psr, cr) on each side. In some cases these results apply generally to different calculi. For example, the inductive step for the case where the cut-formula A is parametric, not principal, on the left is encapsulated in the following theorem where `prop2 mar ?erls ?A (?cl, ?cr)` means that the conclusion of a multicut on `A` with premises `cl` and `cr` is derivable using rules `erls`.

Theorem 2. *If weakening is admissible for the rule set `erls`, all extensions of some rule `(ps, c)` are in the rule set `erls`, and the final rule instance of the left hand subtree is an extension of `(ps, c)` where the cut-formula `A` is not in the succedent of `c` (meaning that `A` is parametric on the left), then* `gen_step2ssr (prop2 mar ?erls) ?A ?sub ?rls (?pscl, ?pscr)` *holds.*

```
mk_sr1 lmg_gen_steps' :
  "[| wk_adm ?erls; extrs {(?ps, ?c)} <= ?erls; ~ ?A :# succ ?c;
        ?pscl = pscmap (extend ?flr) (?ps, ?c) |]
    ==> gen_step2ssr (prop2 mar ?erls) ?A ?sub ?rls (?pscl, ?pscr)"
```

Of course, for a system containing explicit weakening rules, such as GLS, weakening is *a fortiori* admissible. But as noted earlier, we have tried to make our results as general as possible.

This result codifies parametric proof steps, and is quite general enough to apply in the proof for GLS. As we note later, the proof for GLS involves one really difficult case and a lot of fairly routine cases. In dealing with the routine

$$\dfrac{\mu\left\{\dfrac{\Pi_l}{\Box X, X, \Box B \vdash B}\right. }{\Box X \vdash \Box B}\, GLR(\mathrm{B}) \qquad \dfrac{\Pi_r}{\Box B^k, Y \vdash Z}\,(\rho)$$
$$\dfrac{\Box X \vdash \Box B \qquad \Box B^k, Y \vdash Z}{\Box X, Y \vdash Z}\,(multicut)$$

Fig. 1. A multicut on cut formula $\Box B$ where $\Box B$ is left-principal via *GLR*

cases, automated theorem proving has the benefit of ensuring that no detail is overlooked; in addition we have the fact that, as in this example, we often have more general theorems that apply directly to a set of rules such as GLS.

Notice that all of this section has used a shallow embedding of derivations since no explicit derivation trees were required.

6 The Proof of Cut-Admissibility for GLS

Valentini's proof of cut-admissibility for GLS uses a triple induction on the size of the cut-formula, the heights of the derivations of the left and right premises of cut, and a third parameter which he called the "width". Roughly speaking, the width of a cut-instance is the number of *GLR* rule instances above that cut which contain a parametric ancestor of the cut-formula in their conclusion. The Goré and Ramanayake [5] proof follows Valentini in using these three measures, but gives a constructive way to calculate the width of a cut by inspecting the branches of the left and right sub-derivations of a cut rule instance.

As usual, the proof of cut-admissibility for GLS proceeds by considering whether the cut-formula is principal in the left or right premise of the cut, or principal in both. The crux of the proof is a "reduction" when the cut-formula is of the form $\Box B$ and is principal in both the left and right premises of the cut. The solution is to replace this cut on $\Box B$ by cuts which are "smaller" either because their cut-formula is smaller, or because their width is smaller. In reality, most of the work involves a cut instance which is only left-principal as shown in Figure 1, and most of this section is devoted to show how we utilised our general methods to formalise these "reductions" as given by Goré and Ramanayake [5]. But there are some important differences which we now explain.

As explained in §3.2, our general methods do not model derivation trees explicitly, but use a shallow embedding. So our proof uses induction on the size of the cut-formula and on the fact of derivation, rather than on the size of the cut-formula and the height of derivation trees. Also, we actually proved the admissibility of "multicut": that is, if $\Gamma \vdash A^n, \Delta$ and $\Gamma, A^m \vdash \Delta$ are both cut-free derivable, then so is $\Gamma \vdash \Delta$. This avoids problems in the cases where these sequents are derived using contraction on A.

In all other aspects, our proof of cut-admissibility for GLS is based on that given by Goré and Ramanayake [5]. In particular, although we do not actually

```
primrec Der  : "del0 B (Der seq dtl) =
               (if (map conclDT dtl, seq) : glra then
                   if Box B :# antec seq then 1 else 0
                else del0s B dtl)"
        Unf  : "del0 B (Unf dt) = 0"

        Nil  : "del0s B [] = 0"
        Cons : "del0s B (dt # dts) = del0 B dt + del0s B dts"
```

Fig. 2. Definition of function `del0`

require [5, Lemma 19], we use the construction in that lemma, which is fundamental to overcoming the difficulties of adapting standard proofs to GLS, as we explain shortly. Consequently, our proof uses the idea of induction on the *width* as defined in [5], although as we shall see, our proof is expressed in terms of `del0`, which approximates to the ∂^0 of [5], not width *per se*.

To cater for width/`del0`, we could have altered our shallow embedding, `derrec` but that destroys the modularity of our approach. Instead, we defined width/`del0` by using the datatype `dertree` from §3.1 to represent explicit derivation tree objects. These trees, and the general lemmas about them, are similar to the trees of [2]. Thus we "mix and match" a deep embedding of derivation trees with a shallow embedding of inductively defined sets of derivable sequents.

To ensure the correctness of our "mixing and matching" we needed the following relationship between the two embeddings, where a *valid* tree is one whose inferences are in the set of rules and which as a whole has no premises.

Lemma 3. *Sequent `a` is shallowly derivable from the empty set of premises using rules `rls` iff some derivation tree `dt` is valid wrt. `rls` and has a conclusion `a`.*

```
derrec_valid_eq :
"(?a : derrec ?rls {}) = (EX dt. valid ?rls dt & conclDT dt = ?a)"
```

We now define a function `del0` in Figure 2 which is closely related to ∂^0 and the width of a cut of [5], although we are able to avoid mentioning the annotated derivations of [5]. The argument of this function will be the sub-tree μ of Figure 1, `glra` is the set of *all* *GLR*-rule instances, `antec` is a function to return the antecedent of a sequent, and `:#` is multiset membership. Thus, the calculation of `del0` traces up each branch of an explicit derivation tree until an instance of the *GLR* rule is found: it then counts 1 if $\Box B$ is in the antecedent, meaning that B is in the X of the statement of *GLR*. Where the derivation tree branches below any *GLR* rule, the value is summed over the branches.

We now give a sequence of lemmata which eventually give the "reduction" for left-and-right *GLR*-principal occurrences of a cut-formula of the form $\Box B$.

Lemma 4 (`gr19e`)**.** *If μ is a valid derivation tree with conclusion $\Box X, X, \Box B \vdash B$, and `del0` $\mu = 0$, then $\Box X, X \vdash B$ is derivable.*

```
"[| valid glss ?mu ; del0 ?B ?mu = 0 ; (conclDT ?mu) =
      (mset_map Box ?X + ?X + {# Box ?B#} |- {#?B#}) |]
  ==> (mset_map Box ?X + ?X |- {#?B#}) : derrec glss {}" : Thm.thm
```

Proof. By applying the *GLR* rule to the conclusion, we can derive $\Box X \vdash \Box B$. Tracing $\Box B$ upwards in μ, it is parametric in each inference, except possibly weakening, contraction or the axiom $\Box B \vdash \Box B$. That is, since `del0` $\mu = 0$, when we meet a *GLR* inference as we trace upwards, $\Box B$ must have already disappeared (through weakening). So, tracing upwards, we can change each instance of $\Box B$ to $\Box X$ in the usual way. The axiom $\Box B \vdash \Box B$ is changed to $\Box X \vdash \Box B$, which is derivable. Contraction is not problematic since we use, as the inductive hypothesis, that *all* occurrences of $\Box B$ can be replaced by $\Box X$.

To abbreviate the statement of several lemmas, we define a function `muxbn`:

```
defs muxbn_def : "muxbn B n ==
     ALL mu dtb seq. ([conclDT mu], seq) : glr B -->
      del0 B mu <= n --> (Der seq [mu], dtb) : masdt glss (Box B)"
```

Intuitively, `muxbn B n` states that if the left premise derivation consists of a derivation μ whose conclusion can be used as the premise of an instance of the $GLR(B)$ rule, and `del0 B` $\mu \leq n$, then multicut-admissibility for explicit derivation trees (`masdt`) holds for the rule set `glss` with cut-formula $\Box B$: see Figure 1, which is from [5]. That is, the multicut in Figure 1 is admissible.

The next lemma says that multicut admissibility on B implies `muxbn` B 0.

Lemma 5 (`cw0_lbl, cw0_mas'`). *If μ is a valid derivation tree with conclusion $\Box X, X, \Box B \vdash B$, and `del0` B $\mu = 0$, and multicut on B is admissible, and $\Box B^k, Y \vdash Z$ is derivable, then $\Box X, Y \vdash Z$ is derivable.*

Proof. $\Box X \vdash \Box B$ is derivable from $\Box X, X, \Box B \vdash B$ via *GLR*(X,B). By Lemma 4, $\Box X, X \vdash B$ is derivable. The rest of the proof is by induction on the derivation of $\Box B^k, Y \vdash Z$, in effect, by tracing relevant occurrences of $\Box B$ up that derivation. Weakening and contraction involving $\Box B$ are not problematic, and the axiom $\Box B \vdash \Box B$ is changed to $\Box X \vdash \Box B$, which is derivable. Suppose an inference $GLR(Y, C)$ is encountered, where B is in Y. This inference, allowing for multiple copies of B, is as shown below left where Z is Y with B deleted:

$$\dfrac{\Box B^k, B^k, \Box Z, Z, \Box C \vdash C}{\Box B^k, \Box Z \vdash \Box C}\,GLR(\mathrm{Y,C}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\text{Lemma 4}}{\Box X, X \vdash B} \quad \Box X, B^k, \Box Z, Z, \Box C \vdash C}{\Box X, \Box X, X, \Box Z, Z, \Box C \vdash C}\,\mathit{mcut(B)}}{\Box X, X, \Box Z, Z, \Box C \vdash C}\,\mathit{ctr}}{\Box X, \Box Z \vdash \Box C}\,GLR(\mathrm{C})$$

By induction, we have $\Box X, B^k, \Box Z, Z, \Box C \vdash C$ is derivable. From there we have the derivation shown above right. As the machine-checking process confirmed, additional weakening steps are necessary if $\Box B$ is in Z or if B is in $\Box Z$. $\dashv$

This construction is like that of case 2(a)(ii) in the proof of Theorem 20 of [5]. Having shown, in Lemma 5, that `muxbn` B 0 holds, we now use the construction of [5, Lemma 19] to show that `muxbn` $B\ n$ holds for all n: except that our inductive assumptions and results involve admissibility of multicut, not cut. Again we use induction: assume that `muxbn` $B\ n$, and show that `muxbn` $B\ (n+1)$ holds.

So suppose a derivation tree $\mu / \Box X \vdash \Box B$ has a bottom inference $GLR(X, B)$, as shown in Figure 1, and `del0` $B\ \mu = n + 1$. We follow the construction of [5, Lemma 19] to obtain a derivation μ' of $\Box X, X, \Box B \vdash B$, where `del0` $B\ \mu' \leq n$.

Since `del0` $B\ \mu > 0$, the tree μ is as shown below left (with other branches not shown). We first delete the topmost application of GLR(A) leaving a tree μ^-. Then adjoin $\Box A$ to each antecedent of μ^- obtaining the tree on the right, whose topmost sequent is now a weakened axiom, and which requires us to weaken in an occurrence of A just above the final GLR-rule instance:

$$
\begin{array}{cl}
\Box G, G, \Box B^k, B^k, \Box A \vdash A & \\ \hline
\Box G, \Box B^k \vdash \Box A & GLR(\mathrm{A}) \\ \hline
\vdots & \\ \hline
\Box X, X, \Box B \vdash B & \\ \hline
\Box X \vdash \Box B & GLR(\mathrm{X,B})
\end{array}
\qquad\qquad
\begin{array}{cl}
\Box A, \Box G, \Box B^k \vdash \Box A & \\ \hline
\vdots & \\ \hline
\Box A, \Box X, X, \Box B \vdash B & \\ \hline
\Box A, A, \Box X, X, \Box B \vdash B & (\mathit{weakening}) \\ \hline
\Box A, \Box X \vdash \Box B & GLR(\mathrm{B})
\end{array}
$$

The tree on the right has a smaller value of `del0` B, and so can be used as the left branch of an admissible (i.e. "smaller") multicut. We do this twice, the right branches being derivations of $\Box X, X, \Box B \vdash B$ and $\Box G, G, \Box B^k, B^k, \Box A \vdash A$ respectively; which after contractions, give derivations of $\Box A, \Box X, X \vdash B$ and $\Box G, G, \Box X, B^k, \Box A \vdash A$. These two are cuts 1 and 2 of the description in [5, Lemma 19], producing derivations which are cut-free equivalents of Λ_{11} and Λ_{12} from [5, Lemma 19]. The result `gr19a` in the Isabelle code gives the existence of these two derivations; it uses the result `add_Box_ant` which permits adding $\Box A$ to the antecedent of each sequent of a derivation tree.

We combine these derivations of $\Box A, \Box X, X \vdash B$ and $\Box G, G, \Box X, B^k, \Box A \vdash A$ using an admissible ("smaller") multicut on B, and then use contraction to obtain $\Box G, G, \Box A, \Box X, X \vdash A$. This is cut 3 of [5, Lemma 19]. Then the GLR rule gives $\Box G, \Box X \vdash \Box A$. This is derivation Λ_2 of [5, Lemma 19]. Because of this GLR rule at this point, we do not need to worry about the `del0` B values of any of the subtrees mentioned above, whose existence is given by the inductive assumptions of multicut-admissibility. We now weaken the conclusion of this tree to $\Box X, \Box G, \Box B^k \vdash \Box A$, giving (a counterpart of) Λ_3 of [5, Lemma 19].

Returning to μ^-, as below left, we this time adjoin $\Box X$ in the antecedent, giving the tree below right, but we can now use Λ_3 as a derivation for its leaf:

$$
\dfrac{\dfrac{\dfrac{\Box G, \Box B^k \vdash \Box A}{\vdots}}{\Box X, X, \Box B \vdash B}}{\Box X \vdash \Box B}\,GLR
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Lambda_3}{\Box X, \Box G, \Box B^k \vdash \Box A}}{\vdots}}{\Box X, \Box X, X, \Box B \vdash B}\,(contraction)}{\Box X, X, \Box B \vdash B}}{\Box X \vdash \Box B}\,GLR
$$

That is, we have replaced a given derivation μ of $\Box X, X, \Box B \vdash B$ where `del0` $B\ \mu = n+1$, with a derivation μ' of $\Box X, X, \Box B \vdash B$ where `del0` $B\ \mu' = n$.

We record this as the result as Lemma 6 (`gr19b`) below, however, we do not use this result directly. Instead, we obtain Lemma 6 (`gr19c'`) below by referring to Figure 1 and noting that we have replaced μ by μ'.

Lemma 6 (`gr19b, gr19c'`). *Assume that multicut-admissibility (`mas glss`) holds for cut-formula B, and that `muxbn` $B\ n$ holds.*

(a) If μ is a derivation of $\Box X, X, \Box B \vdash B$, where `del0` $B\ \mu = n+1$, then there exists another derivation μ' of $\Box X, X, \Box B \vdash B$ with `del0` $B\ \mu' \leq n$.
(b) `muxbn` $B\ (n+1)$ holds.

```
gr19b : "[| ((conclDT ?mu) =
             (mset_map Box ?X + ?X + {# Box ?B#} |- {#?B#})) ;
            valid glss ?mu ; del0 ?B ?mu = Suc ?n ;
            mas glss ?B = UNIV; muxbn ?B ?n |]
      ==> EX mupr. valid glss mupr &
                  conclDT mupr = conclDT ?mu & del0 ?B mupr <= ?n"
gr19c' : "[| muxbn ?B ?n; mas glss ?B = UNIV |] ==> muxbn ?B (Suc ?n)"
```

Proof. The proof of the first part is the construction described above. For the second part, we are given an instance of Figure 1, where μ has `del0` $B\ \mu = n+1$. Using the first part, we can transform μ to μ', with `del0` $B\ \mu' = n$. Since `muxbn` $B\ n$ holds, the multicut on $\Box B$ where the left premise derivation is $\mu'/\Box X \vdash \Box B$ is admissible. Hence the conclusion $\Box X, Y \vdash Z$ of Figure 1 is derivable.

The next result, which we do not use directly, approximates to [5, Lemma 19].

Lemma 7 (`gr19d`). *If multicut-admissibility holds for cut-formula B, `muxbn` B 0 holds, and $\Box X, X, \Box B \vdash B$ is derivable, then $\Box X, X \vdash B$ is derivable.*

Proof. By using Lemma 6(b) repeatedly, `muxbn` $B\ n$ holds for any n.

Lemma 8 (`cw0_mas, cut_glr`). *If multicut-admissibility (`mas glss`) holds for cut-formula B, then it (`masdt glss`) holds for cut-formula $\Box B$ if the left premise derivation ends with a GLR(B) rule. That is, `muxbn` $B\ n$ holds for any n.*

```
cut_glr : "[| mas glss ?B = UNIV; ([conclDT ?mu], ?seq) : glr ?B |]
      ==> (Der ?seq [?mu], ?dtb) : masdt glss (Box ?B)"

cw0_mas : "mas glss ?B = UNIV ==> muxbn ?B ?n"
```

Proof. By combining Lemma 5 with repeated use of Lemma 6(b).

We now have the admissibility of principal multicuts on $\Box B$, which we indicated was the crux of the cut-admissibility for **GL**. For the other cases, the usual proofs hold. Although there is no particular difficulty in showing this, many details need to be checked: this takes almost 3 pages of [5], even with "omitting details", and citing "standard transformations". The value of automated theorem proving at this point is simply to ensure that all the necessary details have been checked. The proof uses techniques described in §5, which means that a relatively small amount of the proofs remaining for this theorem are peculiar to the GLS calculus. Consequently, we get:

Theorem 3. *Multicut is admissible in GLS.*

```
(?G |- ?D) : derrec glss {} ; (?P |- ?S) : derrec glss {}
 ===> (?G + (ms_delete {?A} ?P) |- (ms_delete {?A} ?D) + ?S))
       : derrec glss {}
```

References

1. J E Dawson and R Goré. Embedding display calculi into logical frameworks: Comparing Twelf and Isabelle. *ENTCS* 42, 89–103.
2. J E Dawson and R Goré. Formalised Cut Admissibility for Display Logic. In Proc. TPHOLS'02, LNCS 2410, 131–147, Springer, 2002.
3. A Gacek. The Abella interactive theorem prover (system description) In Proc. IJCAR 2008 LNCS 5195:154-161, Springer, 2008.
4. R Goré. Invited talk in the Third Indian Conference on Logic and its Applications, Chennai, January 2009.
5. R Goré and R Ramanayake. Valentini's Cut-elimination for Provability Logic Resolved. Proc. AiML 2008, pp 67-86, College Publications `http://users.rsise.anu.edu.au/~rpg/publications.html`
6. D Leivant. On the Proof Theory of the Modal Logic for Arithmetic Provability. Journal of Symbolic Logic, 46:531538, 1981.
7. A Moen. The proposed algorithms for eliminating cuts in the provability calculus GLS do not terminate. NWPT 2001, Norwegian Computing Center, 2001-12-10 `http://publ.nr.no/3411`
8. F Pfenning. Structural cut elimination. In *Proc. LICS 94*, 1994.
9. F Pfenning and C Schrmann. System description: Twelf — a meta-logical framework for deductive systems. In Proc. CADE-16, pages 202–206, Springer-Verlag LNAI 1632.
10. R M Solovay. Provability Interpretations of Modal Logic. Israel Journal of Mathematics, 25:287304, 1976.
11. S Valentini. The Modal Logic of Provability: Cut-elimination. Journal of Philosophical Logic, 12:471476, 1983.