From Display Calculi to Deep Nested Sequent Calculi: Formalised for Full Intuitionistic Linear Logic

Jeremy Dawson, Ranald Clouston, Rajeev Goré, Alwen Tiu

From Display Calculi to Deep Nested Sequent Calculi: Formalised for Full Intuitionistic Linear Logic

Jeremy E. Dawson[1], Ranald Clouston[2], Rajeev Goré[1], and Alwen Tiu[3]

[1] Research School of Computer Science, Australian National University
[2] Department of Computer Science, Aarhus University
[3] School of Computer Engineering, Nanyang Technological University

Abstract. Proof theory for a logic with categorical semantics can be developed by the following methodology: define a sound and complete display calculus for an extension of the logic with additional adjunctions; translate this calculus to a shallow inference nested sequent calculus; translate this calculus to a deep inference nested sequent calculus; then prove this final calculus is sound with respect to the original logic. This complex chain of translations between the different calculi require proofs that are technically intricate and involve a large number of cases, and hence are ideal candidates for formalisation. We present a formalisation of this methodology in the case of Full Intuitionistic Linear Logic (FILL), which is multiplicative intuitionistic linear logic extended with par.

1 Introduction

Belnap's Display Calculus [1] is a powerful modular approach to structural proof theory. Display calculi are often easy to design for a given logic [9] and enjoy a generic algorithm for cut-elimination. However they usually require the logic to be expanded with new structural connectives, raising the question of conservativity, and hence soundness, with respect to the original logic. They also do not enjoy a genuine subformula property and hence are ill-suited to backwards proof search. Various authors [4, 15, 10, 11, 14, 5] have addressed these shortcomings by using some variation of *nested sequent calculus* with *deep inference* [13]. Such deep nested calculi employ a syntax similar to display calculi, but lack their ease of design and generic cut-elimination algorithm. Conversely, deep nested calculi can be designed to have a genuine subformula property, and a "separation property" that trivially yields conservativity results [10, 11, 5]. Since display calculi and deep nested calculi can have contrasting strengths, it is useful to provide sequent calculi in both styles for a given logic. The crux of such a development is the proof of equivalence between the display and deep nested calculi.

Proving the equivalence of display and deep nested calculi is technically intricate and can involve the verification of hundreds of cases. Such proofs proceed via an intermediate calculus, a *shallow* inference nested sequent calculus, and it is the proof of the equivalence of shallow and deep calculi that is the most

demanding, requiring that every possible interaction of shallow and deep proof rules be covered. We hence have a fruitful proof theoretic methodology which cries out both for mechanised proof checking to increase confidence in its results, and for the use of automated tactics to reduce the drudgery of attaining them. We describe such a formalisation for Full Intuitionistic Linear Logic (FILL) [12], following our earlier work on display and deep nested calculi for this logic [5].

Schellinx [16] considered the standard multiple-conclusioned sequent calculus for intuitionistic logic (where the right-implication rule is restricted to prevent collapse to classical logic) without weakening and contraction, and showed that it does not enjoy cut-elimination. Hyland and de Paiva [12] gave this logic (with cut) the name Full Intuitionistic Linear Logic, and defined categorical semantics for it, giving several natural examples of categories exhibiting the required structure. They further claimed to have found a cut-free sequent calculus for FILL, in which term-assignments on formulae are put to novel use to block unsound applications of right-implication, via a freeness check on abstracted variables. Reasoning about freeness in the presence of binders is a well known source of subtle error, and a major topic of formalisation research (e.g. [19]). Indeed Bierman [2] found a counter-example to Hyland and de Paiva's cut-elimination proof exploiting a binding-related error, and presented two solutions using even more complex type-annotations, one due to Bellin. Braüner and de Paiva [3] subsequently suggested a cut-free calculus relying on annotations on sequents, rather than formulae. Two previous claims in the literature to annotation-free sequent calculi for FILL were erroneous, as discussed in [5].

Our recent contribution [5] to this rather vexed history was to show that annotations are not necessary; we gave a sound and complete display calculus for FILL and showed how it can be compiled into two equivalent nested sequent calculi, one with shallow inference and the other with deep inference. In particular the deep calculus is cut-free complete for FILL, enjoys the sub-formula property, and supports terminating backward proof-search, from which we obtained the NP-completeness of the validity problem for FILL. The derivation of these results, given in more detail in [6], is unavoidably highly technical, and given its difficulty and the history of FILL outlined above we sought to formalise our results in the proof assistant Isabelle/HOL. The completed formalisation presented in this paper finally establishes the correctness of a sequent calculus for this logic. In fact an initial attempt to prove the soundness of our calculus was found to be flawed only when we tried to formalise it (see below for more details), so this development has been an invaluable part of even our 'pen-and-paper' work.

We now outline our proof stategy [5] as formalised in this paper. FILL has the usual relation between multiplicative conjunction $\otimes$ and implication $\multimap$, where $\rightarrow$ denotes an arrow in a category (see §3):

$$(A \otimes B) \rightarrow C \text{ iff } (B \otimes A) \rightarrow C \text{ iff } A \rightarrow (B \multimap C) \tag{1}$$

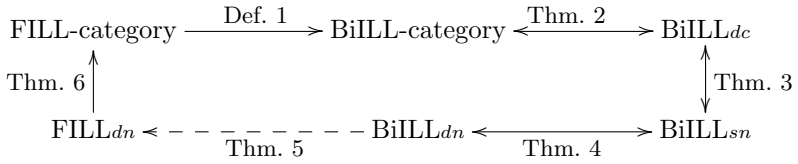
The *display property*, which underlies the generic cut-elimination algorithm of display calculi, requires the introduction of new structural connectives, so it is clarifying to regard a display calculus as defining a larger logic with new logical

connectives. In this case we have a multipicative *exclusion* $-\!\!<$, defined with respect to multiplicative disjunction (or *par*) $\parr$ in a manner dual to (1):

$$C \to (A \parr B) \text{ iff } C \to (B \parr A) \text{ iff } (C -\!\!< B) \to A \tag{2}$$

In §2 and §3, we extend the syntax of FILL with $-\!\!<$ and extend the semantics of FILL to obtain Bi-Intuitionistic Linear Logic (BiILL). In §4 we give a display calculus BiILL$_{dc}$ which is easily seen to be sound and complete for BiILL and hence complete for the sublogic FILL. The soundness of BiILL$_{dc}$ for FILL corresponds to the conservativity of BiILL over FILL. We first attempted to prove the soundness result directly via a rewriting strategy which removed occurrences of exclusion from a BiILL$_{dc}$-derivation of a FILL-formula to give an exclusion-free BiILL$_{dc}$-derivation of the same FILL-formula. This rewriting strategy turned out to be flawed, as it may not always terminate. Instead, we define two nested sequent calculi for BiILL: BiILL$_{sn}$ with shallow inference in §5, and BiILL$_{dn}$ with deep inference in §6. The equivalence of BiILL$_{sn}$ and BiILL$_{dn}$, established as Thm. 4 in §6, is the technical highlight of the formalisation, with 616 cases verified. Thm. 5 in §7 shows that because of a *separation* property, BiILL$_{dn}$ easily specialises to a deep nested calculus FILL$_{dn}$ with no trace of exclusion. Thm. 6 then shows that the calculus FILL$_{dn}$ is sound for FILL, thereby proving conservativity of BiILL over FILL. §8 concludes.

Our methodology is summarised below, where a solid arrow indicates that every valid formula in the source is also valid in the target, and a dashed arrow represents the same notion restricted to FILL formulae only:



We used Isabelle/HOL 2005 so that we could rework the cut-elimination proofs from our previous work on formalising cut elimination for display calculi [7]. As discussed in §5, one problem we found was the lack of support for nested datatypes involving multisets. The Isabelle/HOL theory files for our formalisation are at: `http://users.cecs.anu.edu.au/~jeremy/isabelle/2005/fill`

2 Formulae, sequents and derivations

We explain briefly the data structures we use to encode formulae, structures, sequents and derivations. The language of formulae for BiILL is defined using the grammar below where p denotes a propositional variable:

$$A ::= p \mid I \mid \bot \mid A \otimes A \mid A \parr A \mid A \multimap A \mid A -\!\!< A$$

A FILL formula is just a BiILL formula without any occurrences of $-\!\!<$.

BiILL formulae are defined formally in Isabelle as follows. We let the type variable `'s` be (Isabelle) strings.

```
datatype 's pformula =
    Btimes pformula pformula ("_ &&& _" [68,68] 67)
  | Bplus pformula  pformula ("_ +++ _" [66,66] 65)
  | Blolli pformula pformula ("_ --o _" [64,64] 63)
  | Bexcl pformula pformula ("_ --< _" [64,64] 63)
  | Btrue ("T") | Bfalse("F") | FV string | PP string
```

Here the binary constructors correspond to, respectively, $\otimes$, $\parr$, $\multimap$ and $\mathbin{-\!\!<}$, the unary constructors `Btrue` and `Bfalse` encode the units I and $\bot$ respectively, the constructor `PP` is used to encode propositional variables, and `FV` is used to encode "scheme variables"; we shall come back to these shortly.

We define the immediate (proper) subformula relation, `ipsubfml`.

```
ipsubfml     :: "('a pformula * 'a pformula) set"
inductive "ipsubfml" (* proper immediate subformula relation *)
  intrs ips_and1 "(P, P &&& Q) : ipsubfml" (etc)
```

A BiILL-sequent is a pair $X_a \vdash X_s$ of an *antecedent* and a *succedent* structure, defined respectively as follows:

$$X_a ::= A \mid \Phi \mid X_a, X_a \mid X_a < X_s \qquad\qquad X_s ::= A \mid \Phi \mid X_s, X_s \mid X_a > X_s$$

where Φ is a structural constant. A FILL-sequent is a BiILL-sequent containing no occurrence of $<$ or $\mathbin{-\!\!<}$. The relation between formulae and structures will be made precise in the next section.

Structures are represented by the datatype below:

```
datatype 's pstructr = Comma ('s pstructr) ('s pstructr)
  | Gt ('s pstructr) ('s pstructr) | Lt ('s pstructr) ('s pstructr)
  | Phi | Structform ('s pformula) | SV 's
```

where `Comma`, `Gt`, `Lt` and `Phi` correspond to the structural connectives ',', $>$, $<$ and Φ. The operator `Structform` casts a formula into a structure. The constructor `SV` represents scheme variables for structures. Since we must reason about arbitrary derivations, we have to allow derivations to contain structure variables and reason about their instantiations. We do not encode explicitly the notion of antecedent/succeedent structures in the data type; these notions are enforced via separate predicates when needed (see e.g. §5).

Sequents and rules of the calculus are represented by

```
datatype 'a sequent = Sequent 'a 'a
types 'a psc = "'a list * 'a" (* single step inference *)
types 'a rule = 'a sequent psc
```

The premises of a rule are represented using a list of sequents while the conclusion is a single sequent. Thus `(prems, concl)` means a rule with premises `prems` and conclusion `concl`. We write `$X |- $Y` to denote `(Sequent X Y)`.

We now briefly describe the functions we used to encode derivability. A fuller account is given in [8]. 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 or sequents, but we will refer to them as formulae.

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

An inference rule `(ps, c) : '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 formulae derivable using rules `rls` from the set `prems` of premises. These were defined as inductive sets, using auxiliary functions `dersl` and `dersrec`, which concern the derivability of all members of a list. So to say $(\texttt{ps}, \texttt{c}) \in \texttt{derl rls}$ reflects the shape of a derivation tree: `ps` is a list of exactly the premises used, in the correct order, whereas $\texttt{c} \in \texttt{derrec rls prems}$ holds even for any set of premises `prems` containing those required.

Since we use cut-admissibility for Display Calculi, and also some rules of BiILL$_{sn}$ are *admissible* (not *derivable*) in BiILL$_{dn}$ (see Theorem 4), we we need to formalise the notion that a rule of one system is *admissible* in another system: (ps, c) is admissible iff: if all premises in ps are derivable, then c is derivable:

$$(ps, c) \in \texttt{adm rls} \iff (\mathit{set}\ ps \subseteq \texttt{derrec rls} \ \Rightarrow \ c \in \texttt{derrec rls})$$

3 Formalising Categorial Semantics

Definition 1. *A* FILL-category *is a category equipped with*

- *a symmetric monoidal closed structure* $(\otimes, I, \multimap)$
- *a symmetric monoidal structure* $(\parr, \bot)$
- *a natural family of* weak distributivity *arrows* $A \otimes (B \parr C) \to (A \otimes B) \parr C$.

A BiILL-category *is a FILL-category where the* $\parr$ *bifunctor has a* co-closure $-\!\!<$, *so there is a natural isomorphism between arrows* $A \to B \parr C$ *and* $A -\!\!< B \to C$.

To interpret BiILL sequents in the category semantics, we use the following translation from (antecedent/succeedent) structures to formulae:

	A	Φ	X, Y	$X > Y$	$X < Y$
τ^a	A	I	$\tau^a(X) \otimes \tau^a(Y)$		$\tau^a(X) -\!\!< \tau^s(Y)$
τ^s	A	$\bot$	$\tau^s(X) \parr \tau^s(Y)$	$\tau^a(X) \multimap \tau^s(Y)$	

Definition 2. *A FILL- (resp. BiILL-) sequent* $X \vdash Y$ *is* satisfied *by a FILL- (resp. BiILL-) category if, given any valuation of its propositional variables as objects, there exists an arrow* $I \to \tau^a(X) \multimap \tau^s(Y)$. *It is FILL- (resp. BiILL-)* valid *if it is satisfied by all such categories.*

To establish validity it suffices to show a hom-set is non-empty inthe *free* FILL- (resp. BiILL-) categories. We sketch the definitions of these below, omitting the *equations* that hold between arrows in these categories.

Category: $A \xrightarrow{id} A \qquad A \xrightarrow{f' \circ f} A''$

Symmetric Monoidal: $A \heartsuit B \xrightarrow{f \heartsuit g} A' \heartsuit C \qquad (A \heartsuit B) \heartsuit C \underset{\alpha^{-1}}{\overset{\alpha}{\rightleftarrows}} A \heartsuit (B \heartsuit C)$

$$K \heartsuit A \underset{\lambda^{-1}}{\overset{\lambda}{\rightleftarrows}} A \qquad A \heartsuit K \underset{\rho^{-1}}{\overset{\rho}{\rightleftarrows}} A \qquad A \heartsuit B \xrightarrow{\gamma} B \heartsuit A$$

Closed:

$$A \multimap B \xrightarrow{A \multimap g} A \multimap C \qquad (A \multimap B) \otimes A \xrightarrow{\varepsilon} B \qquad A \xrightarrow{\eta} B \multimap A \otimes B$$

Weak Distributivity: $A \otimes (A' \parr A'') \xrightarrow{\omega} (A \otimes A') \parr A''$

Co-Closed:

$$A \mathrel{-\!\!<} B \xrightarrow{f \mathrel{-\!\!<} B} A' \mathrel{-\!\!<} B \qquad A \parr B \mathrel{-\!\!<} A \xrightarrow{\varepsilon} B \qquad A \xrightarrow{\eta} B \parr (A \mathrel{-\!\!<} B)$$

Fig. 1. Arrows of the free BiILL-category

Definition 3. *The* free *FILL- (resp. BiILL-) category has FILL- (resp. BiILL) formulae as objects, and, given objects A, A, A'', B, C and arrows $f : A \to A', f' : A' \to A'', g : B \to C$, and $(\heartsuit, K) \in \{(\otimes, I), (\parr, \bot)\}$, has arrows as in Fig. 1, where the co-closure arrows exist in the free BiILL-category only.*

Def. 3 can also be written as a deducibility relation between formulae where we use the turnstile $\vdash$ to assert the existence of an arrow, without specifying how it is constructed. For example, the rules below capture the definition of 'closed' above, in which the finer details of the construction $A \multimap g$ are elided:

$$\frac{B \vdash C}{A \multimap B \vdash A \multimap C} \qquad \frac{}{(A \multimap B) \otimes A \vdash B} \qquad \frac{}{A \vdash B \multimap (A \otimes B)}$$

We formalise this deducibility relation using the sequents of the previous section, but with only a formula (not a more complex structure) on each side. Then the identity arrow and composition of arrows become the usual identity and cut rules. As another example, the "closure" rules above are encoded as:

```
lolli_monoR == (["B" |- "C"], "A" --o "B" |- "A" --o "C")
lolliD == ([], ("A" --o "B") &&& "A" |- "B")
lolliI == ([], "A" |- "B" --o ("A" &&& "B"))
```

Our encoding is faithful because each arrow required by Def. 3 is encoded as one such rule giving the rules `biill_cat_rules` with subset `fill_cat_rules`.

4 Formalising Display Calculi

Our formalisation of the display calculus BiILL$_{dc}$ is very similar to that in [7], so we shall not give full details here. The display system BiILL$_{dc}$ is given in Fig. 2, where double-lined inference rules are invertible.

Cut and identity:

$$\text{(id)}\quad p \vdash p \qquad \text{(cut)}\ \frac{X \vdash A \qquad A \vdash Y}{X \vdash Y}$$

Logical rules:

$$(I \vdash)\ \frac{\Phi \vdash X}{I \vdash X} \qquad (\bot \vdash)\quad \bot \vdash \Phi \qquad (\vdash I)\quad \Phi \vdash I \qquad (\vdash \bot)\ \frac{X \vdash \Phi}{X \vdash \bot}$$

$$(\otimes \vdash)\ \frac{A, B \vdash X}{A \otimes B \vdash X} \qquad (\vdash \otimes)\ \frac{X \vdash A \qquad Y \vdash B}{X, Y \vdash A \otimes B}$$

$$(\parr \vdash)\ \frac{A \vdash X \qquad B \vdash Y}{A \parr B \vdash X, Y} \qquad (\vdash \parr)\ \frac{X \vdash A, B}{X \vdash A \parr B}$$

$$(\multimap \vdash)\ \frac{X \vdash A \qquad B \vdash Y}{A \multimap B \vdash X > Y} \qquad (\vdash \multimap)\ \frac{X \vdash A > B}{X \vdash A \multimap B}$$

$$(\prec \vdash)\ \frac{A < B \vdash X}{A \prec B \vdash X} \qquad (\vdash \prec)\ \frac{X \vdash A \qquad B \vdash Y}{X < Y \vdash A \prec B}$$

Structural rules:

$$\text{(rp)}\ \frac{X \vdash Y > Z}{X, Y \vdash Z} \quad \text{(rp)}\ \frac{X, Y \vdash Z}{Y \vdash X > Z} \quad \text{(drp)}\ \frac{X < Y \vdash Z}{X \vdash Y, Z} \quad \text{(drp)}\ \frac{X \vdash Y, Z}{X < Z \vdash Y}$$

$$(\Phi \vdash)\ \frac{X, \Phi \vdash Y}{X \vdash Y} \quad (\vdash \Phi)\ \frac{X \vdash \Phi, Y}{X \vdash Y} \quad \text{(Com} \vdash)\ \frac{X, Y \vdash Z}{Y, X \vdash Z} \quad (\vdash \text{Com})\ \frac{X \vdash Y, Z}{X \vdash Z, Y}$$

$$\text{(Ass} \vdash)\ \frac{W, (X, Y) \vdash Z}{(W, X), Y \vdash Z} \qquad (\vdash \text{Ass})\ \frac{W \vdash (X, Y), Z}{W \vdash X, (Y, Z)}$$

$$\text{(Grnb} \vdash)\ \frac{W, (X < Y) \vdash Z}{(W, X) < Y \vdash Z} \qquad (\vdash \text{Grnb})\ \frac{W \vdash (X > Y), Z}{W \vdash X > (Y, Z)}$$

Fig. 2. Display calculus BiILL$_{dc}$ for BiILL

As in our previous formalisation [7], structure variables like X are encoded as `$"X"` and formula variables like A are encoded as `"A"`. The quotes are necessary since we handle substitutions explicitly rather than via Isabelle variables [7]. For example, the cut rule is encoded as:

```
cutr == ([($"X" |- "A"), ("A" |- $"Y")], ($"X" |- $"Y"))
```

To prove the cut-admissibility result we largely reuse the code we used to prove cut-admissibility for the display calculus for relation algebras [7]. Belnap [1] gave eight conditions, C1-C8, which guarantee cut-elimination for a given display calculus. Previous work has shown that all except C8 are trivial or can be checked automatically [7]. The proof that a connective satisfies Belnap's C8 condition has to be coded in part individually for each connective, but, even so, we were able to reuse most of our previous code. Each display logic rule of BiILL$_{dc}$ is encoded as shown above for cut, giving the set of rules named `biilldc`, and its strict subset `biilldc_cf` which excludes the cut rule.

Given a set `S` of encoded rules, the set `rulefs S` is the (infinite) set of substitutional instances of members of `S`. From our previous work [8], the rule `(ps, c)` with list of premise (sequents) `ps` and conclusion (sequent) `c` is an admissible rule of `S` if the following holds, where colon is set-membership $\in$:

```
(?ps : dersrec ?rls {} --> ?c : derrec ?rls {}) ==> (?ps, ?c) : adm ?rls
```

Cut and identity: $\dfrac{}{p \Rightarrow p}\ id \qquad \dfrac{\mathcal{S} \Rightarrow \mathcal{U}, A \quad A, \mathcal{V} \Rightarrow \mathcal{T}}{\mathcal{S}, \mathcal{V} \Rightarrow \mathcal{U}, \mathcal{T}}\ cut$

Structural rules:

$$\frac{\mathcal{S} \Rightarrow \mathcal{T}, \mathcal{T}'}{(\mathcal{S} \Rightarrow \mathcal{T}) \Rightarrow \mathcal{T}'}\ drp_1 \quad \frac{\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{T}'}{\mathcal{S} \Rightarrow (\mathcal{T} \Rightarrow \mathcal{T}')}\ rp_1 \quad \frac{(\mathcal{S} \Rightarrow \mathcal{S}'), \mathcal{T} \Rightarrow \mathcal{T}'}{(\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{S}') \Rightarrow \mathcal{T}'}\ gl$$

$$\frac{(\mathcal{S} \Rightarrow \mathcal{T}) \Rightarrow \mathcal{T}'}{\mathcal{S} \Rightarrow \mathcal{T}, \mathcal{T}'}\ drp_2 \quad \frac{\mathcal{S} \Rightarrow (\mathcal{T} \Rightarrow \mathcal{T}')}{\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{T}'}\ rp_2 \quad \frac{\mathcal{S} \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}}{\mathcal{S} \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}', \mathcal{T})}\ gr$$

Logical rules: $\dfrac{}{\bot \Rightarrow \cdot}\ \bot_l \qquad \dfrac{\mathcal{S} \Rightarrow \mathcal{T}}{\mathcal{S} \Rightarrow \mathcal{T}, \bot}\ \bot_r \qquad \dfrac{\mathcal{S} \Rightarrow \mathcal{T}}{\mathcal{S}, \mathrm{I} \Rightarrow \mathcal{T}}\ \mathrm{I}_l \qquad \dfrac{}{\cdot \Rightarrow \mathrm{I}}\ \mathrm{I}_r$

$$\frac{\mathcal{S}, A, B \Rightarrow \mathcal{T}}{\mathcal{S}, A \otimes B \Rightarrow \mathcal{T}}\ \otimes_l \qquad \frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}' \Rightarrow B, \mathcal{T}'}{\mathcal{S}, \mathcal{S}' \Rightarrow A \otimes B, \mathcal{T}, \mathcal{T}'}\ \otimes_r$$

$$\frac{\mathcal{S}, A \Rightarrow \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}', A \parr B \Rightarrow \mathcal{T}, \mathcal{T}'}\ \parr_l \qquad \frac{\mathcal{S} \Rightarrow A, B, \mathcal{T}}{\mathcal{S} \Rightarrow A \parr B, \mathcal{T}}\ \parr_r$$

$$\frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}', A \multimap B \Rightarrow \mathcal{T}, \mathcal{T}'}\ \multimap_l \qquad \frac{\mathcal{S} \Rightarrow \mathcal{T}, (A \Rightarrow B)}{\mathcal{S} \Rightarrow \mathcal{T}, A \multimap B}\ \multimap_r$$

$$\frac{\mathcal{S}, (A \Rightarrow B) \Rightarrow \mathcal{T}}{\mathcal{S}, A \prec\!\!- B \Rightarrow \mathcal{T}}\ \prec\!\!-_l \qquad \frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}' \Rightarrow A \prec\!\!- B, \mathcal{T}, \mathcal{T}'}\ \prec\!\!-_r$$

Fig. 3. The shallow inference system BiILL$_{sn}$.

We then proved cut-admissibility as below, where colon now simply states the name of the theorem. The formal proof reuses the work described in [7].

Theorem 1 (Cut-Admissibility). *From cut-free* BiILL$_{dc}$*-derivations of* $X \vdash A$ *and* $A \vdash Y$ *we can obtain a cut-free* BiILL$_{dc}$*-derivation of* $X \vdash Y$*.*

```
dc_cut_adm : "rulefs {cutr} <= adm (rulefs biilldc_cf)"
```

We can now gain our first result linking proof theory and semantics. In the following, `(?A |- ?B)` is a sequent with arbitrary formulae on each side. Arbitrary structures would appear as `($?A |- $?B)` [7].

Theorem 2. BiILL$_{dc}$ *is* sound *and* cut-free complete *for BiILL-validity (where the appellation* `cf` *captures cut-free).*

```
dc_cat_equiv_cf : "((?A |- ?B) : derrec (rulefs biilldc_cf) {}) =
  ((?A |- ?B) : derrec (rulefs biill_cat_rules) {})"
```

5 Shallow Nested Sequent Calculi

In [5] nested sequents are defined as below, where A_i and B_j are formulae:

$$S \;\; T ::= S_1, \ldots, S_k, A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n, T_1, \ldots, T_l$$

We use Γ and Δ for multisets of formulae and use P, Q, S, T, X, Y, etc., for nested sequents, and $\mathcal{S}$, $\mathcal{X}$, etc., for multisets of nested sequents and formulae. The empty multiset is $\cdot$ ('dot'). A nested sequent is essentially a display structure, but with the associativity and commutativity of the comma structural connective implicit in the use of multisets. The sequent arrow $\Rightarrow$ overloads both $>$ and $<$, depending on whether it occurs in an antecedent or a succeedent position in the sequent. This overloading simplifies the presentation of the nested sequent rules [5]. The shallow inference system BiILL$_{sn}$ for BiILL is given in Fig. 3.

The most faithful encoding of a nested sequent would be one that uses multisets as a datatype, which is supported by recent versions of Isabelle [18]. However due to incompatibilities between versions of Isabelle we have been constrained to use an older version of Isabelle to allow us to reuse proofs for display calculi developed in that version [7]. Our definition of nested sequents is thus as below:

```
datatype nested = NComma nested nested | Nseq nested nested
                | NPhi | NStructform formula | NSV string
```

In our definition, `NSeq` is the nested sequent turnstile $\Rightarrow$, `NComma` is the comma of nested sequent calculi and `NPhi` is its unit. As for display calculi, we allow `=>` instead of `Nseq` and `,,,` instead of `NComma`. In our Isabelle formalisation, BiILL$_{sn}$ rules are prefixed by `sn`, e.g., the *rp* rule is named `sn_rp`. The entire set of rules is called `biillsn` and its cut-free subset is `biillsn_cf` (see file `N_Rls.thy`). As with the display calculus, we define a function `nrulefs` to generate the (infinite) set of all substitution instances of a given rule.

We defined a relation `ms_deep_equiv` of *multiset-equivalence*, under which any two Isabelle nested sequents are equivalent if they would be the same if a collection of Isabelle nested sequents, separated by commas, were considered as a multiset. This includes where the difference between two Isabelle nested sequents occurs at any depth. Its definition relies on a function `ms_of_ns` for "multiset of Isabelle nested sequent" which turns (eg) the Isabelle nested sequent $(S,T),U$ into the multiset `{# S, T, U #}`, and a relation `ms_ms_deep_equiv`, which expresses equivalence of multisets of Isabelle nested sequents.

To prove BiILL$_{dc}$ and BiILL$_{sn}$ equivalent, we define translation functions

```
consts nested_to_str :: "bool => nested => structr"
       nested_to_seq :: "nested => structr sequent"
       seq_to_nested :: "structr sequent => nested"
       str_to_nested :: "structr => nested"
```

where the translation from a nested sequent to a display calculus structure depends on whether it is in an antecedent or succedent position.

The first two of these functions convert a nested sequent to a display calculus sequent or structure by converting '$\Rightarrow$' to '$\vdash$' (for `nested_to_seq`), to '$>$' (for `nested_to_str True`), or to '$<$' (for `nested_to_str False`), and converting comma to comma. The latter two convert '$\vdash$', '$>$', and '$<$' to '$\Rightarrow$'.

For example `nested_to_seq` takes $(A \Rightarrow B) \Rightarrow (C \Rightarrow D)$ to $A < B \vdash C > D$, and `seq_to_nested` does the reverse.

Considering the set of display calculus sequents with $<$ and $>$ only in antecedent and succedent positions respectively (expressed as `seq_LtGtOK`), and the

set of nested sequents, then we have mutually inverse bijections `nested_to_seq` and `seq_to_nested` between these sets. We can express this by:

```
nest_seq_equiv : "(seq_LtGtOK ?seq & seq_to_nested ?seq = ?nes) =
      ((EX a s. ?nes = ($a => $s)) & nested_to_seq ?nes = ?seq)"
```

The proof systems BiILL$_{dc}$ and BiILL$_{sn}$ are very similar; their structural rules are the same (modulo some notational variance). The only difference is that BiILL$_{dc}$ requires that logical rules be applied only to a 'displayed' formula, i.e. the principal formula must appear in isolation either on the left or on the right of the turnstile. Their equivalence is not surprising, so we state their equivalence here and refer the reader to the proof scripts for details.

Theorem 3. *The display sequent $A \vdash B$ is cut-free* BiILL$_{dc}$*-derivable iff the nested sequent $A \Rightarrow B$ is cut-free* BiILL$_{sn}$*-derivable, and $\mathrm{I} \vdash A$ is cut-free* BiILL$_{dc}$*-derivable iff the nested sequent $\cdot \Rightarrow A$ is cut-free* BiILL$_{sn}$*-derivable.*

```
dc_sn_equiv_alt = "((?A |- ?B) : derrec (rulefs biilldc_cf) {}) =
         ((?A => ?B) : derrec (nrulefs biillsn_cf) {})"
dc_sn_equiv : "((T |- ?A) : derrec (rulefs biilldc_cf) {}) =
         (($NPhi => ?A) : derrec (nrulefs biillsn_cf) {})"
```

6 Deep Nested Sequent Calculi

Deep inference rules for nested sequents are applied in a *context*, i.e., a nested sequent with a hole [ ]. We use several notions of contexts in our formalisation. The first two accept a set of nested sequent rules and return a set of nested sequent rules while the third accepts and returns a set of Isabelle nested sequents

```
ctxt :: "nested psc set => nested psc set"
dctxt :: "nested psc set => nested psc set"
hctxt :: "nested set => nested set"
```

For example, if $([P], C) \in R$, where $[P]$ is a singleton list (rather than a context) containing one premise, then $([X[P]], X[C]) \in$ `ctxt` R is also a single premise rule. Likewise, if $C \in R$ and $X[\,]$ is a hollow context then $X[C] \in$ `hctxt` R.

Some of the proofs involving `ctxt` were easier using a related definition `dctxt` where $X[S \Rightarrow T]$ means adding nested sequents to S or to T, rather than to $S \Rightarrow T$. For example, if $([P_1 \Rightarrow P_2], (C_1 \Rightarrow C_2)) \in$ `dctxt` R, then $([P_1, X \Rightarrow P_2], (C_1, X \Rightarrow C_2)) \in$ `dctxt` R. Similarly, if $([P_1 \Rightarrow P_2], (C_1 \Rightarrow C_2)) \in$ `dctxt` R then $([(P_1 \Rightarrow P_2) \Rightarrow X], ((C_1 \Rightarrow C_2) \Rightarrow X)) \in$ `dctxt` R

The nested sequent system BiILL$_{dn}$ is given in Fig. 4. Notice that it lacks the structural rules. The zero-premise rules require that certain sequents or contexts are *hollow*, i.e., contain no occurrences of formulae. The branching rules require operations to merge contexts and nested sequents, which are explained below.

The *merge set* $X_1 \bullet X_2$ of two sequents X_1 and X_2 is defined as:

$$
\begin{aligned}
X_1 \bullet X_2 = \{\ & (\Gamma_1, \Gamma_2, Y_1, \ldots, Y_m \Rightarrow \Delta_1, \Delta_2, Z_1, \ldots, Z_n) \ \mid \\
& X_1 = (\Gamma_1, P_1, \ldots, P_m \Rightarrow \Delta_1, Q_1, \ldots, Q_n) \ \text{and} \\
& X_2 = (\Gamma_2, S_1, \ldots, S_m \Rightarrow \Delta_2, T_1, \ldots, T_n) \ \text{and} \\
& Y_i \in P_i \bullet S_i \text{ for } 1 \leq i \leq m \text{ and } Z_j \in Q_j \bullet T_j \text{ for } 1 \leq j \leq n \ \}
\end{aligned}
$$

Propagation rules:

$$\frac{X[\mathcal{S} \Rightarrow (A, \mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}]}{X[\mathcal{S}, A \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}]}\ pl_1 \qquad \frac{X[(\mathcal{S} \Rightarrow \mathcal{T}, A), \mathcal{S}' \Rightarrow \mathcal{T}']}{X[(\mathcal{S} \Rightarrow \mathcal{T}), \mathcal{S}' \Rightarrow A, \mathcal{T}']}\ pr_1$$

$$\frac{X[\mathcal{S}, A, (\mathcal{S}' \Rightarrow \mathcal{T}') \Rightarrow \mathcal{T}]}{X[\mathcal{S}, (\mathcal{S}', A \Rightarrow \mathcal{T}') \Rightarrow \mathcal{T}]}\ pl_2 \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}, A, (\mathcal{S}' \Rightarrow \mathcal{T}')]}{X[\mathcal{S} \Rightarrow \mathcal{T}, (\mathcal{S}' \Rightarrow \mathcal{T}', A)]}\ pr_2$$

Identity and logical rules: In branching rules, $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$, $\mathcal{S} \in \mathcal{S}_1 \bullet \mathcal{S}_2$ and $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$.

$$\frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\mathcal{U}, p \Rightarrow p, \mathcal{V}]}\ id^d \qquad \frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\bot, \mathcal{U} \Rightarrow \mathcal{V}]}\ \bot_l^d \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}]}{X[\mathcal{S} \Rightarrow \mathcal{T}, \bot]}\ \bot_r^d$$

$$\frac{X[\mathcal{S} \Rightarrow \mathcal{T}]}{X[\mathcal{S}, \mathrm{I} \Rightarrow \mathcal{T}]}\ \mathrm{I}_l^d \qquad \frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\mathcal{U} \Rightarrow \mathrm{I}, \mathcal{V}]}\ \mathrm{I}_r^d$$

$$\frac{X[\mathcal{S}, A, B \Rightarrow \mathcal{T}]}{X[\mathcal{S}, A \otimes B \Rightarrow \mathcal{T}]}\ \otimes_l^d \qquad \frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2 \Rightarrow B, \mathcal{T}_2]}{X[\mathcal{S} \Rightarrow A \otimes B, \mathcal{T}]}\ \otimes_r^d$$

$$\frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S}, A \multimap B \Rightarrow \mathcal{T}]}\ \multimap_l^d \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}, (A \Rightarrow B)]}{X[\mathcal{S} \Rightarrow \mathcal{T}, A \multimap B]}\ \multimap_r^d$$

$$\frac{X_1[\mathcal{S}_1, A \Rightarrow \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S}, A \parr B \Rightarrow \mathcal{T}]}\ \parr_l^d \qquad \frac{X[\mathcal{S} \Rightarrow A, B, \mathcal{T}]}{X[\mathcal{S} \Rightarrow A \parr B, \mathcal{T}]}\ \parr_r^d$$

$$\frac{X[\mathcal{S}, (A \Rightarrow B) \Rightarrow \mathcal{T}]}{X[\mathcal{S}, A \prec B \Rightarrow \mathcal{T}]}\ \prec_l^d \qquad \frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S} \Rightarrow A \prec B, \mathcal{T}]}\ \prec_r^d$$

Fig. 4. The deep inference system BiILL$_{dn}$.

The merge set $X_1[\ ] \bullet X_2[\ ]$ of two contexts $X_1[\ ]$ and $X_2[\ ]$ is defined in Figure 5. If $X[\ ] = X_1[\ ] \bullet X_2[\ ]$ we say $X_1[\ ]$ and $X_2[\ ]$ are a *partition* of $X[\ ]$. We extend the notion of a merge set between multisets of formulae and sequents as follows. Given $\mathcal{X} = \Gamma \cup \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \Delta \cup \{Y_1, \ldots, Y_n\}$ their merge set contains all multisets of the form: $\Gamma \cup \Delta \cup \{Z_1, \ldots, Z_n\}$ where $Z_i \in X_i \bullet Y_i$.

In Isabelle we defined merged sequents using triples, so in effect $(X_1, X_2, X) \in$ `rmerge` means X is a sequent in $X_1 \bullet X_2$. This is easier to define than in the paper where multisets are used, because we require simply that each structural atom (formula or structure variable) in X is replaced by Φ in exactly one of X_1 or X_2. That is, each atom has to go in one partition or the other. Likewise, to express the idea of $X_1[\ ] \bullet X_2[\ ]$ we define $(X_1, X_2, X) \in$ `rmerge1`(Y_1, Y_2, Y) to be similar except that at one spot X contains Y, where X_i contains Y_i.

We illustrate here some key steps in the formalisation of the equivalence between BiILL$_{sn}$ and BiILL$_{dn}$. We show only the translation from shallow nested sequent proofs to deep nested sequent proofs, which is the more difficult part of the equivalence. The key lemma here is that the rules rp_i, drp_i, gl and gr, which are in BiILL$_{sn}$ but not in the deep calculus BiILL$_{dn}$, are admissible in BiILL$_{dn}$.

$$
\begin{array}{ll}
\text{If } X_1[\,] = [\,] \text{ and } X_2[\,] = [\,] \text{ then } X_1[\,] \bullet X_2[\,] = \{[\,]\} \\
\text{If } X_1[\,] & = (\Gamma_1, Y_1[\,], P_1, \ldots, P_m \Rightarrow \Delta_1, Q_1, \ldots, Q_n) \text{ and} \\
\quad X_2[\,] & = (\Gamma_2, Y_2[\,], S_1, \ldots, S_m \Rightarrow \Delta_2, T_1, \ldots, T_n) \text{ then} \\
X_1[\,] \bullet X_2[\,] & = \{\ (\Gamma_1, \Gamma_2, Y[\,], U_1, \ldots, U_m \Rightarrow \Delta_1, \Delta_2, V_1, \ldots, V_n) \quad | \\
& \quad\ Y[\,] \in Y_1[\,] \bullet Y_2[\,] \text{ and } U_i \in P_i \bullet S_i \text{ for } 1 \leq i \leq m \text{ and} \\
& \quad\ V_j \in Q_j \bullet T_j \text{ for } 1 \leq j \leq n\ \} \\
\text{If } X_1[\,] & = (\Gamma_1, P_1, \ldots, P_m \Rightarrow \Delta_1, Y_1[\,], Q_1, \ldots, Q_n) \text{ and} \\
\quad X_2[\,] & = (\Gamma_2, S_1, \ldots, S_m \Rightarrow \Delta_2, Y_2[\,], T_1, \ldots, T_n) \text{ then} \\
X_1[\,] \bullet X_2[\,] & = \{\ (\Gamma_1, \Gamma_2, U_1, \ldots, U_m \Rightarrow \Delta_1, \Delta_2, Y[\,], V_1, \ldots, V_n) \mid \\
& \quad\ Y[\,] \in Y_1[\,] \bullet Y_2[\,] \text{ and } U_i \in P_i \bullet S_i \text{ for } 1 \leq i \leq m \text{ and} \\
& \quad\ V_j \in Q_j \bullet T_j \text{ for } 1 \leq j \leq n\ \}
\end{array}
$$

Fig. 5. Merging of contexts

Lemma 1. *The rules drp_1, rp_1, drp_2, rp_2, gl, and gr permute up over all logical rules of* BiILL$_{dn}$.

This is the permutation lemma, that (in general) where one of the shallow rules in question follows a rule of BiILL$_{dn}$ in a derivation, then the derivation can be re-ordered so that the shallow rule precedes the deep rule. We illustrate here a step in the proof of this permutation lemma, i.e., when permuting structural rules over a single-premise logical rule. We proved theorems of the following form

```
p_irp_anda : "?c = ($?ca => $?cs) -->
  ([?p], ?c) : ctxt (nrulefs {sn_anda}) -->
  (?c, ?c') : ms_deep_equiv --> ([?c'], ?d') : nrulefs {invert sn_rp} -->
  (EX p' q q' d. (?p, p') : ms_deep_equiv &
    ([p'], q) : nrulefs {invert sn_rp} & (q, q') : ms_deep_equiv &
    ([q'], d) : dctxt (nrulefs {sn_anda}) & (d, ?d') : ms_deep_equiv)"
```

That is, where a structural rule, e.g., rp_1, appears below a deep logical rule in a derivation, the derivation steps may be permuted so that the logical rule follows the other rule. It may be noted that this result uses `dctxt`, not `ctxt`, in the conclusion. This made semi-automatic proof easier; the lemmas for all the logical rules concerned (6 of them) for all the structural rules involved (6 of them) were done using just three separate sets of tactics.

The proofs of the permutation lemma involved a large number of cases, because a sequent expression such as $X[\mathcal{S} \Rightarrow \mathcal{T}]$ can match a given sequent Z in numerous ways, for two reasons:

- for multisets $\mathcal{S}$ and $\mathcal{T}$, there can be multiple ways to achieve $(\mathcal{S}, \ldots) \in$ `ms_deep_equiv`
- the size of context $X[\,]$ is arbitrary, so $\mathcal{S} \Rightarrow \mathcal{T}$ can match any part of Z.

The attempted proof encounters many obviously impossible cases such as a formula matching $\mathcal{S} \Rightarrow \mathcal{T}$. After these are eliminated, we counted the cases where a goal (such as the conclusion of `p_rp_anda`) is actually solved. These cases numbered 616, which shows the value of automating the process as much as possible.

Theorem 4. *All rules cut-free derivable in* BiILL$_{sn}$ *are (cut-free) admissible in* BiILL$_{dn}$*.*

```
sn_dn_der : "derl (nrulefs biillsn_cf) <= adm biilldn"
```

A sequent is cut-free provable (derivable from the empty set of assumption sequents) in BiILL$_{sn}$ *iff it is provable* BiILL$_{dn}$*.*

```
sn_dn_equiv : "(?r : derrec (nrulefs biillsn_cf) {}) =
               (?r : derrec biilldn {})"
```

The formula A is cut-free provable in the display calculus iff it is (cut-free) provable in BiILL$_{dn}$*.*

```
dc_dn_equiv : "((T |- ?A) : derrec (rulefs biilldc_cf) {}) =
               (($NPhi => ?A) : derrec biilldn {})"
```

7 Soundness of the Deep Nested Calculus FILL$_{dn}$

Definition 4. *A nested sequent is a nested* FILL-sequent *if it has no nesting of sequents on the left of $\Rightarrow$, and no occurrences of $-\!\!<$.*

BiILL$_{dn}$ enjoys the *separation* property that rule applications with FILL-sequents as their conclusions may only have FILL-sequents as their premises; note that the display calculus BiILL$_{dc}$ obviously lacks this property, given (drp). We hence define FILL$_{dn}$ as the proof system obtained from BiILL$_{dn}$ by restricting to FILL-sequents and removing the unnecessary rules pr_1, pl_2, $-\!\!<^d_l$ and $-\!\!<^d_r$. Our goal here is to show that FILL$_{dn}$ is sound with respect to FILL categories.

The formula translation of τ^s (see §3) can be adapted straightforwardly to map (nested) FILL-sequents to FILL-formulae. Such a sequent S is FILL-valid if there is an arrow $I \to \tau^s(S)$ in the free FILL-category.

In our formalisation, we in fact defined the rules `filldn` of FILL$_{dn}$ without requiring the sequents involved to be FILL-sequents. We then defined a corresponding set `sfilldn` of rules, requiring that the sequents are FILL-sequents.

Theorem 5. *A* BiILL$_{dn}$*-derivation of a* FILL$_{dn}$*-sequent is a* FILL$_{dn}$*-derivation.*

```
dn_der_biill_sfill : "[| ?c : derrec biilldn ?ps;
    ALL U. (U, ntau True ?c) : ipsubfml^* --> U ~: excl_fmls |] ==>
    ?c : derrec sfilldn ?ps"
```

The soundness proof consists of a series of lemmas showing that the rules of FILL$_{dn}$ preserves validity going downward (from premises to conclusion). We illustrate one particularly challenging lemma that involves context merging.

Lemma 2. *Take $X[\,] \in X_1[\,] \bullet X_2[\,]$ and $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$. Then the following arrows exist in the free FILL-category for all A, B, Γ_1 and Γ_2:*

(a) $\tau^s(X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2 \Rightarrow B, \mathcal{T}_2]) \;\to\; \tau^s(X[\Gamma_1, \Gamma_2 \Rightarrow A \otimes B, \mathcal{T}])$*;*
(b) $\tau^s(X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]) \;\to\; \tau^s(X[\Gamma_1, \Gamma_2, A \multimap B \Rightarrow \mathcal{T}])$*;*
(c) $\tau^s(X_1[\Gamma_1, A \Rightarrow \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]) \;\to\; \tau^s(X[\Gamma_1, \Gamma_2, A \parr B \Rightarrow \mathcal{T}])$*;*

This lemma corresponds to soundness of branching logical rules of FILL$_{dn}$. We proved the inductive part (involving the contexts X_i and X) once, in the form of `fill_rmerge1_der_nseq`. Then we proved the base case (without X_i and X) for each connective, resulting in three theorems `lem27s`, of which one is shown.

```
fill_rmerge1_der_nseq : "[| (?Ta, ?Tb, ?Tc) : rmerge1 (?A, ?B, ?C);
  ?prems = {ntau True ?A &&& ntau True ?B |- ntau True ?C};
  (?W, ntau False ?C) : ipsubfml^* & ?W : excl_fmls;
  ALL U. (U, ntau True ?Tc) : ipsubfml^* --> U ~: excl_fmls |] ==>
  (ntau True ?Ta &&& ntau True ?Tb |- ntau True ?Tc) :
    derrec (rulefs fill_cat_rules) ?prems"
lem27s (first one) :
  "[| (?A, ?B, ?C) : dn_ands; (?Ta, ?Tb, ?Tc) : rmerge1 (?A, ?B, ?C);
    ALL U. (U, ntau True ?Tc) : ipsubfml^* --> U ~: excl_fmls |] ==>
  (ntau True ?Ta &&& ntau True ?Tb |- ntau True ?Tc) :
    derrec (rulefs fill_cat_rules) {}"
```

Theorem 6. *For every rule of* FILL$_{dn}$*, if the premises are FILL-valid then so is the conclusion.*

```
filldn_rules_valid : "[| (?ps, ?c) : filldn;
  ALL U. (U, ntau True ?c) : ipsubfml^* --> U ~: excl_fmls |] ==>
  (T |- ntau True ?c) : derrec (rulefs fill_cat_rules)
    ((%p. T |- ntau True p) ` set ?ps)"
```

Theorem 7. *A formula is* FILL*-valid iff it is* FILL$_{dn}$*-provable, and BiILL is conservative over FILL.*

8 Conclusion and future work

Finding a cut-free sequent calculus for FILL has been a notoriously difficult problem, as we have reviewed in our introduction, and involved candidate proof systems that turned out to be incomplete. Our formalisation finally establishes convincingly that our deep nested calculus FILL$_{dn}$ is both sound and complete for FILL. Apart from our FILL$_{dn}$, all other existing proof calculi for FILL still require complex annotations to ensure cut-elimination.

The formalisation and verification described here was a significant task: it was the major activity for an experienced Isabelle user (Dawson) for about seven months, not including some months more working on the proof which ultimately was found to be flawed (see §1). and not counting the proof of Thm 1, reused from [7]. The most difficult single part of it was the proof of Lemma 1, discussed in §6. The difficulty in defining a nested sequent datatype containing multisets of nested sequents (see §5) was also significant. The value of the formal verification is clear since it led us to find the flaw in the previous attempt at a proof.

Taking a broader perspective, we have shown a detailed formalisation of a methodology for deriving a deep nested sequent calculus for a logic from its categorical semantics via a display calculus and a shallow nested sequent calculus for a natural extension containing additional connectives. For future work, we

plan to apply this same formalised methodology to derive deep nested sequent calculi for a wide range of logics [4, 15, 10, 11, 17, 14]. A difference between these logics and FILL is the use of "additive" context splitting in branching rules, where contexts are duplicated across premises. In the presence of contraction rules, our multiplicative context splitting can simulate such additive splitting, just as in the traditional sequent calculus. That is, one can apply contraction to duplicate every formula occurrence in the context before splitting them. Thus we think a similar formalisation effort for, say, nested sequent calculi for modal logics [11], would benefit significantly from our current formalisation.

References

1. N. D. Belnap. Display logic. *J. Philos. Logic*, 11:375–417, 1982.
2. G. M. Bierman. A note on full intuitionistic linear logic. *Ann. Pure Appl. Logic*, 79(3):281–287, 1996.
3. T. Bräuner and V. de Paiva. A formulation of linear logic based on dependency-relations. In *CSL*, pages 129–148, 1997.
4. K. Brünnler. Deep sequent systems for modal logic. *Arch. Math. Logic*, 48(6):551–577, 2009.
5. R. Clouston, J. E. Dawson, R. Goré, and A. Tiu. Annotation-free sequent calculi for full intuitionistic linear logic. In *CSL*, pages 197–214, 2013.
6. R. Clouston, J. E. Dawson, R. Goré, and A. Tiu. Annotation-free sequent calculi for full intuitionistic linear logic - extended version. *CoRR*, abs/1307.0289, 2013.
7. J. E. Dawson and R. Goré. Formalised cut admissibility for display logic. In *TPHOLs*, pages 131–147, 2002.
8. J. E. Dawson and R. Goré. Generic methods for formalising sequent calculi applied to provability logic. In *LPAR*, pages 263–277, 2010.
9. R. Goré. Substructural logics on display. *Log. J. IGPL*, 6(3):451–504, 1998.
10. R. Goré, L. Postniece, and A. Tiu. Cut-elimination and proof search for bi-intuitionistic tense logic. In *AiML*, pages 156–177, 2010.
11. R. Goré, L. Postniece, and A. Tiu. On the correspondence between display postulates and deep inference in nested sequent calculi for tense logics. *LMCS*, 7(2), 2011.
12. M. Hyland and V. de Paiva. Full intuitionistic linear logic (extended abstract). *Ann. Pure Appl. Logic*, 64(3):273–291, 1993.
13. R. Kashima. Cut-free sequent calculi for some tense logics. *Studia Log.*, 53:119–135, 1994.
14. J. Park, J. Seo, and S. Park. A theorem prover for boolean BI. In *POPL*, pages 219–232, 2013.
15. F. Poggiolesi. The method of tree-hypersequents for modal propositional logic. In *Trends in Logic IV*, pages 31–51, 2009.
16. H. Schellinx. Some syntactical observations on linear logic. *J. Logic Comput.*, 1(4):537–559, 1991.
17. L. Straßburger. Cut elimination in nested sequents for intuitionistic modal logics. In *FoSSaCS*, pages 209–224, 2013.
18. D. Traytel, A. Popescu, and J. C. Blanchette. Foundational, compositional (co)datatypes for higher-order logic: Category theory applied to theorem proving. In *LICS*, pages 596–605, 2012.
19. C. Urban. Nominal techniques in isabelle/HOL. *J. Automat. Reason.*, 40(4):327–356, 2008.