GENERALITY AND EXISTENCE 1: QUANTIFICATION AND FREE LOGIC

Greg Restall*
Philosophy Department
The University of Melbourne
restall@unimelb.edu.au

OCTOBER 15, 2015

Version 0.95

Abstract: In this paper, I motivate a cut free sequent calculus for classical logic with first order quantification, allowing for singular terms free of existential import. Along the way, I motivate a criterion for rules designed to answer Prior's question about what distinguishes rules for logical concepts, like 'conjunction' from apparently similar rules for putative concepts like 'tonk', and I show that the rules for the quantifiers—and the existence predicate—satisfy that condition.

1 SEQUENTS AND DEFINING RULES

Let's take it for granted for the moment that learning a language involves—at least in part—learning how assertions and denials expressed in that language bear on one another. The basic connection, of course, is that to assert A and to deny A *clash*. When we learn conjunction, we learn that there is a clash involved in asserting A, asserting B and denying $A \wedge B$. Similarly, when we learn disjunction, we learn that there is a clash involved in asserting $A \vee B$, denying A and denying B.

One way to systematically take account of the kinds of clashes involved in these acts of assertion and denial is through the language of the sequent calculus. Given collections Γ and Δ of sentences from our language $\mathcal{L}$, a *sequent* $\Gamma \succ \Delta$ makes the claim that there is a clash involved in asserting each element of Γ and denying each

*This work has been in progress for quite some time. I am grateful for audiences at the weekly Logic Seminar and the weekly Philosophy Seminar at the University of Melbourne, as well as presentations in Aberdeen, the Australian National University, Sun Yat-Sen University, St Andrews, LMU Munich, and the Australasian Assocation for Logic Conference for helpful comments on this material. I am grateful to Aldo Antonelli, Conrad Asmus, Bogdan Dicher, Allen Hazen, Lloyd Humberstone, Catarina Dutilh Novaes, Andrew Parisi, Graham Priest, Dave Ripley, Gillian Russell, Jeremy Seligman, Shawn Standefer and Crispin Wright for discussions on these topics, and especially to Dave Ripley and Shawn Standefer for detailed comments on drafts of this paper. ¶ I dedicate this paper to Aldo Antonelli, whose untimely death leaves us all the poorer. ¶ This research is supported by the Australian Research Council, through Grant DP150103801. ¶ A current draft of this paper is available at http://consequently.org/writing/generality-and-existence-1/.

element of Δ. The structural rules of the sequent calculus can be understood in the following way [9]. *Identity*—

$$A \succ A$$

—there is a clash involved in asserting A and denying A. *Weakening*—

$$\frac{\Gamma \succ \Delta}{\Gamma, A \succ \Delta} \qquad \frac{\Gamma \succ \Delta}{\Gamma \succ A, \Delta}$$

—if there is a clash involved in a position, it remains after adding an extra assertion or a denial. *Contraction*—

$$\frac{\Gamma, A, A \succ \Delta}{\Gamma, A \succ \Delta} \qquad \frac{\Gamma \succ A, A, \Delta}{\Gamma \succ A, \Delta}$$

—the number of times a claim is asserted or denied is irrelevant to the presence or absence of a clash. If there is a clash when A is asserted [or denied] $n + 1$ times it would have been present if it had been asserted [or denied] n times. *Cut*—

$$\frac{\Gamma \succ A, \Delta \quad \Gamma, A \succ \Delta}{\Gamma \succ \Delta}$$

—if there is a clash involved in asserting A (in concert with asserting Γ and denying Δ) and a clash involved in denying A (in that same context) then there is a clash in the underlying context.

I will not *defend* these rules here (I have done so elsewhere [9]), except by showing how they may be *used* to shed light on the behaviour of rules as definitions of logical concepts, and in particular the scope for rules concerning quantification and existence.

So, let's suppose we have a language $\mathcal{L}$ governed by some relation $\succ_{\mathcal{L}}$ satisfying the constrains of *Identity*, *Weakening*, *Contraction* and *Cut*. How might we extend the language with a new concept? One way to characterise this task is to characterise the new language $\mathcal{L}'$ (extending $\mathcal{L}$) with a new relation $\succ_{\mathcal{L}'}$ extending $\succ_{\mathcal{L}}$. Suppose, for example, we have a language consisting of some atomic vocabulary, together with the one place operator $\neg$ satisfying the usual classical principles. Suppose we wish to extend the language with a binary connective for conjunction. One way to do this is to impose a *Left* and a *Right* rule for conjunction from the sequent calculus, something like this pair of rules—

$$\frac{\Gamma, A, B \succ \Delta}{\Gamma, A \wedge B \succ \Delta}\,[\wedge L] \qquad \frac{\Gamma \succ A, \Delta \quad \Gamma \succ B, \Delta}{\Gamma \succ A \wedge B, \Delta}\,[\wedge R]$$

—and this would certainly have the desired effect. However, it seems hard to motivate such an addition, or to explain how these rules succeed in introducing a concept while a superficially similar pair of rules for a connective like '*tonk*' [8]—

$$\frac{\Gamma, B \succ \Delta}{\Gamma, A \mathit{\ tonk\ } B \succ \Delta}\,[\mathit{tonk}L] \qquad \frac{\Gamma \succ A, \Delta}{\Gamma \succ A \mathit{\ tonk\ } B, \Delta}\,[\mathit{tonk}R]$$

—do not seem to do quite the same kind of job. Imposing an arbitrary pair of Left and Right rules is too strong for the successful addition of a new concept, not because it leads to *Bad Things* (like *tonk*) but because it makes claims on $\succ_{\mathcal{L}'}$ that the relation may not be able to meet. Under the assmumptions concerning norms governing assertion and denial we have set out, the rule [*tonkL*] makes a claim concerning assertions of *tonk*-statements, while the rule [*tonkR*] makes a claim concerning denials of *tonk*-statements. An assertion of A *tonk* B involves a clash when an assertion of B involves a clash. Similarly, a denial of A *tonk* B involves a clash when an denial of A involves a clash. The only way that could work, in the presence of *Cut*, *Identity* and *Weakening* is when all positions clash. If the original relation $\succ_{\mathcal{L}}$ was non-trivial, then the new language $\mathcal{L}'$ with its relation $\succ_{\mathcal{L}'}$ is a *revisionary* extension of $\mathcal{L}$, not only adding claims about clashes involving the new vocabulary, but revising our view of what clashes there might be in $\mathcal{L}$. It is not so much an extension to $\mathcal{L}$ as a revision of it.

How did this take place? The rules of *Cut* and *Identity* already connect norms governing assertions and norms governing denials. The rules [*tonkL*] and [*tonkR*] say too much, in making independent and ill-fitting impositions on assertion and on denial. They are ill-suited to be understood as instructions to extend the relation $\succ_{\mathcal{L}}$ on the language $\mathcal{L}$ to a relation $\succ_{\mathcal{L}'}$ on language $\mathcal{L}'$ satisfying *Cut* and *Identity*.

On the other hand, in another sense a pair of Left and Right rules may say *too little*, since they need not specify precisely when assertions (or denials) of *tonk* statements involve a clash—they just say that under conditions there is a clash—and there may be clashes under other conditions too.[1] The Left and Right rules of a sequent calculus can, under certain circumstances, be neither too heavy (like the rules for *tonk*) nor too light (like the rules for *tink*) to pick out a single concept. Nuel Belnap characterised the situation by distinguishing those rules that are not too heavy (they provide a *conservative extension* to the source language $\mathcal{L}$) nor too light (they are *uniquely defining* over that language $\mathcal{L}$) [3]. In the rest of this section I will offer an explanation of how such rules can arise out of a more fundamental criterion, the DEFINING RULE. Here is an example, a defining rule for conjunction:

$$\frac{\Gamma, A, B \succ \Delta}{\Gamma, A \wedge B \succ \Delta}\,[\wedge Df]$$

This differs from a pair of Left and Right rules in two ways. In this case, it char-

[1] Consider this pair of rules for the putative connective *tink*:

$$\frac{\Gamma, A \succ \Delta \quad \Gamma, B \succ \Delta}{\Gamma, A \mathit{\ tink\ } B \succ \Delta}\,[\mathit{tinkL}] \qquad \frac{\Gamma \succ A, \Delta \quad \Gamma \succ B, \Delta}{\Gamma \succ A \mathit{\ tink\ } B, \Delta}\,[\mathit{tinkR}]$$

These rules tell us that an assertion of A *tink* B clashes where an assertion of A and an assertion of B would have clashed, and a denial of A *tink* B clashes where the denial of A and the denial of B would have clashed. These conditions are satisfied if A *tink* B is identified with $A \wedge B$, or with $A \vee B$, or with any of the multitude of other propositions entailed by $A \wedge B$ and entailing $A \vee B$. These rules do too little to characterise a connective.

acterises the behaviour of conjunction formulas on one side (in this case, the left side) of the sequent alone, and second, it is an *invertible* rule, to be read from top-to-bottom and also from bottom-to-top. (This is the function of the double horizontal line.) Understood in terms of assertion and denial, it states that an assertion of a conjunction $A \wedge B$ involves a clash (with the assertions Γ and denials Δ) if and only if the assertions of A and B (with Γ and Δ) also involves a clash.

The $[\wedge Df]$ rule completely characterises clashes involving assertions of conjunctions (alongside the rest of the vocabulary) in terms of clashes involving their components. By itself, it says nothing concerning denials of conjunctions. However, the denials are not left out—the characterisation of denials of conjunctions is achieved by way of the rules of *Identity* and *Cut* in the extended language. If we have some language $\mathcal{L}$ with clashes charactrised in that vocabulary by the relation $\succ_{\mathcal{L}}$, satisfying *Identity* and *Cut* (and perhaps some other rules allowing for composition of arbitrary formulas) then we can consider adding $[\wedge Df]$ to characterise negation, in a new language $\mathcal{L} + \wedge$ with a new consequence relation $\succ_{\mathcal{L}+\wedge}$, constrained to still satisfy *Identity* and *Cut*. In the extended language we have, by *Identity*

$$\frac{}{A \wedge B \succ A \wedge B}\,[Id]$$

(since the assertion of $A \wedge B$ clashes with its denial), and we can then apply $[\wedge Df]$ to this identity sequent to infer *something* concerning denials of conjunctions:

$$\dfrac{\dfrac{}{A \wedge B \succ A \wedge B}\,[Id]}{A, B \succ A \wedge B}\,[\wedge Df]$$

We learn that a denial of a conjunction clashes with the assertion of the conjuncts. This goes *some* way to characterising denials of conjuncts, but we would like to know more—in particular, we would like to know whether the denial of a conjunction clashes in other contexts. Here, we can use the *Cut* rule.

$$\dfrac{\Gamma \succ A, \Delta \qquad \dfrac{\Gamma \succ B, \Delta \qquad \dfrac{\dfrac{}{A \wedge B \succ A \wedge B}\,[Id]}{A, B \succ A \wedge B}\,[\wedge Df]}{\Gamma, A \succ A \wedge B, \Delta}\,[Cut]}{\Gamma \succ A \wedge B, \Delta}\,[Cut]$$

Which gives us a $[\wedge R]$ rule of the expected form:

$$\frac{\Gamma \succ A, \Delta \quad \Gamma \succ B, \Delta}{\Gamma \succ A \wedge B, \Delta}\,[\wedge R]$$

This tells us that denying a conjunction $A \wedge B$ invovles a clash when denying either conjunct involves a clash.

Defining rules can be given for disjunction, the material conditional and negation in the same way:

$$\frac{\Gamma \succ A, B, \Delta}{\Gamma \succ A \vee B, \Delta}\,[\vee Df] \qquad \frac{\Gamma, A \succ B, \Delta}{\Gamma \succ A \supset B, \Delta}\,[\supset Df] \qquad \frac{\Gamma \succ A, \Delta}{\Gamma, \neg A \succ \Delta}\,[\neg Df]$$

In each case, the top-to-bottom reading of the defining rule provides one of the traditional sequent calculus Left/Right rules, and the other can be recovered using *Identity* (on a primitive sequent involving the introduced concept alone), a *Defining Rule* to unwrap the connective on the side on which it is defined, and *Cuts* to make the context general. Here is the case for disjunction:

$$\dfrac{\dfrac{\dfrac{}{A \vee B \succ A \vee B}\,[Id]}{A \vee B \succ A, B}\,[\vee Df] \qquad \Gamma, A \succ \Delta}{\Gamma, A \vee B \succ B, \Delta}\,[Cut] \qquad \Gamma, B \succ \Delta}{\Gamma, A \vee B \succ \Delta}\,[Cut]$$

This gives us the usual rule:

$$\frac{\Gamma, A \succ \Delta \quad \Gamma, B \succ \Delta}{\Gamma, A \vee B \succ \Delta}\,[\vee L]$$

Here is the case for the conditional:

$$\dfrac{\Gamma \succ A, \Delta \qquad \dfrac{\dfrac{\dfrac{}{A \supset B \succ A \supset B}\,[Id]}{A \supset B, A \succ B}\,[\vee Df] \qquad \Gamma, B \succ \Delta}{\Gamma, A \supset B, A \succ \Delta}\,[Cut]}{\Gamma, A \supset B \succ \Delta}\,[Cut]$$

This gives us the usual rule

$$\frac{\Gamma \succ A, \Delta \quad \Gamma, B \succ \Delta}{\Gamma, A \supset B \succ \Delta}\,[\supset L]$$

And finally, for negation:

$$\dfrac{\dfrac{\dfrac{}{\neg A \succ \neg A}\,[Id]}{\succ A, \neg A}\,[\neg Df] \qquad \Gamma, A \succ \Delta}{\Gamma \succ \neg A, \Delta}\,[Cut] \qquad \text{gives us} \qquad \frac{\Gamma, A \succ \Delta}{\Gamma \succ \neg A, \Delta}\,[\neg R]$$

No such justification can be given for *tonk* or *tink*. If we take *tonkL* to be a defining rule:

$$\frac{\Gamma, B \succ \Delta}{\Gamma, A \mathit{\ tonk}\ B \succ \Delta}\,[tonkDf?]$$

then we have defined A *tonk* B to be equivalent to B, and [*tonkR*] is not satisfied by the connective. If we take *tonkR* to be the defining rule:

$$\frac{\Gamma \succ A, \Delta}{\Gamma \succ A \mathit{tonk} B, \Delta} \; [\mathit{tonkDf?}]$$

then we have defined A *tonk* B to be equivalent to A, and [*tonkL*] is not satisfied by the connective. There is no way to introduce something satisfying both *tonk* rules by a defining rule of this kind.[2]

So, defining rules give us an independently motivated answer to Prior's original question concerning when inference rules successfully introduce a new concept. Defining rules introduce a concept to a language, characterising the norms governing assertions (or denials) featuring that concept. Concepts introduced in this way In the next section, we will consider how we can extend the work of defining rules to consider quantifiers.

2 GENERALITY AND CLASSICAL QUANTIFIERS

Conjunction, disjunction, negation and the material conditional are propositional connectives that rely on no assumptions about the internal structure of sentences. Quantifiers are different. For quantifiers, we need our sentences to have some internal structure—in particular, for first order quantifiers, we need to identify a class of singular terms, so we can articulate the connections between claims like these:

$$Fa \quad (\forall x)Fx \quad (\exists x)Fx \qquad Rab \quad (\exists x)Rax \quad (\forall y)(\exists x)Rxy$$

So, let's suppose that our language has a designated class of *singular terms* (we'll use $m, n, \ldots, s, t, \ldots$ for singular terms of various kinds[3]), and a class of *variables* (we'll use $x, y, z, \ldots$ for variables) for the quantifiers $(\forall x), (\forall y), \ldots$ and $(\exists x), (\exists y), \ldots$ In general, if a language allows for sentences to include variables unbound by any quantifier, then the variables will be among the class of singular terms. We will not make that assumption here, and for clarity, we will take it that all variables occuring in sentences must be bound by quantifiers, though nothing important hangs on this.

For each sentence A in our language, we may single out some instances of a singular term occuring in A by enclosing that term parentheses. $A(n)$ is a sentence with some number of instances of n singled out.[4] Given $A(n)$, the formula

[2] I have not here provided a proof that there is no defining rule for *tonk* (whatever that would mean), just that the rules given by Prior cannot do the job of a defining rule for a concept satisfying both Left and Right rules. As a matter of fact, there is no defining rule that will introduce a connective satisfying both *tonk* rules, over a non-trivial consequence relation satisfying *Identity* and *Cut*, if the extension is to keep *Identity* and *Cut*. Defining rules result in conservative extensions, and a consequence relation satisfying *Identity*, *Cut* and the *tonk* rules is trivial.

[3] A mnemonic: 'n' for 'name' and 't' for 'term'. The distinction between names and terms is not important now, but will become important soon.

[4] Yes, that number can be zero.

$(\forall x)A(x)$ is found by replacing those designated instances of n by the variable x and binding the formula with the quantifier $(\forall x)$. So, for example, If $A(n)$ is the formula

$$(Lm\underline{n} \supset L\underline{n}m) \wedge Lnm'$$

with the designated instances of n indicated by underlining, then the corresponding formula $(\exists x)A(n)$ is

$$(\exists x)((Lmx \supset Lxm) \wedge Lnm')$$

This means that we can replace some number instances of a singular term in a formula by variables, and bind them with a quantifier in order to construct a new formula.[5] This leads to the natural question. What is the connection between $A(n)$ and $(\forall x)A(x)$? Or between $A(n)$ and $(\exists x)A(x)$? The classical behaviour of the quantifiers suggests the following pair of defining rules:

$$\frac{\Gamma \succ A(n), \Delta}{\Gamma \succ (\forall x)A(x), \Delta}\,[\forall Df] \qquad \frac{\Gamma, A(n) \succ \Delta}{\Gamma, (\exists x)A(x) \succ \Delta}\,[\exists Df]$$

(where n is not present in the bottom sequent of both rules). These rules are satisfied by the quantifiers in classical first order predicate logic. $(\forall x)A(x)$ is false in a model if and only if we can assign a value for n, for which $A(n)$ is false in that model (provided that n is free to interpret however we wish). On the other hand $(\exists x)A(x)$ is true in a model if and only if we can assign a value for n such that $A(n)$ is true in that model.

These rules can be motivated in terms of assertion and denial, too. Denying $(\forall x)A(x)$ (in the context of asserting Γ and denying Δ) involves a clash if and only if denying $A(n)$ would involve a clash, where n is a name free of any commitments. Asserting $(\exists x)A(x)$ (in the context of asserting Γ and denying Δ) involves a clash if and only if asserting $A(n)$ would involve a clash, where n is a new name.

For these rules to work in the intended way, the name n has to be appropriate. Not every singular term in every language can do the job. Here is an example. Consider RA the finite set of axioms of Robinson's Arithmetic.[6] In the context of classical predicate logic, we can derive $RA \succ 0 \neq 3088$. There is a clash involved in asserting the axioms of RA and in denying that 0 is unequal to 3088. However, the term 3088 does not appear in the axioms of RA. Nonetheless, it would be a mistake to generalise using $[\forall Df]$ to deduce $PA \succ (\forall x)(0 \neq x)$. Although 3088 does not appear in the axioms of RA, it is not logically independent of them. In the syntax of RA, 3088 is a *function* term,[7] which means that it is not free to be interpreted arbitrarily, given the commitments made in the other assertions and denials we have made—in this case, in RA.

[5] We allow for languages in which variables themselves count as singular terms, and languages in which they never occur in formulas unbound, and are not themselves proper singular terms.

[6] In fact, we don't need all of the axioms in RA. The single axiom $(\forall x)(0 \neq x')$ stating that 0 is not the successor of any number will do.

[7] Literally, it is the term '0' with the successor function applied to it 3088 times.

In general, in a given language $\mathcal{L}$ with a consequence relation $\succ$, we will say that a term α in a category $\mathfrak{A}$ DEDUCTIVELY GENERAL iff for each sequent $\Gamma \succ \Delta$ that can be derived, so can $\Gamma[\alpha := \beta] \succ \Delta[\alpha := \beta]$ where α is globally replaced in that sequent by another term β in the category $\mathfrak{A}$. In first order classical predicate logic, *function* terms are not deductively general singular terms, but primitive names are. If we consider the conseqence relation of *Peano Arithmetic*, defined by setting $\Gamma \succ_{\mathsf{PA}} \Delta$ iff $\mathsf{PA}, \Gamma \succ \Delta$ over the language of predicate logic with 0, successor, addition, multiplication and identity, where PA is an axiomatisation of PA, then the constant term 0 is not deductively general, since we have $(\exists x)(0 \neq x') \succ_{\mathsf{PA}}$ (it's inconsistent with the axioms of PA for 0 be the successor of some number), but we do not have $(\exists x)(0' \neq x') \succ_{\mathsf{PA}}$.

Our proposed defining rules $[\forall Df]$ and $[\exists Df]$ make sense only when we impose the restriction that the terms n appearing in the defining rule are deductively general. However, there is a tension between this condition and the form of the defining rules themselves. Consider a toy example, in which we have a language with a primitive predicate F, a stock of names n, m,..., and a single one-place function symbol g, and it is interpreted in the usual classical fashion. In this language, n and m are names, while $g(n)$, $g(m)$, $g(g(n))$, etc., are terms but not names. How can we derive $(\forall x)Fx \succ Fg(n)$ when we extend the language using $[\forall Df]$? We can certainly derive $(\forall x)Fx \succ Fn$ for each name n, as follows:

$$\dfrac{\dfrac{}{(\forall x)Fx \succ (\forall x)Fx}\,[Id]}{(\forall x)Fx \succ Fn}\,[\forall Df]$$

However, there is no way to apply the rule $[\forall Df]$ to generate the conclusion $Fg(n)$, since $g(n)$ is not a name—it is not deductively general. If we were to require that in the extended language, the names remained deductively general, we would require that the consequence relation satisfy the condition of *Specification*:

$$\dfrac{\Gamma \succ \Delta}{\Gamma[n := t] \succ \Delta[n := t]}\,[Spec^{n}_{t}]$$

which permits a gobal replacement of the deductively general term n by the term t of the same category, which may be less general.[8] In other words, the specification rule ensures that the terms n indeed are deductively general. If we could appeal to the specification rule in a derivation, we could conclude $(\forall x)Fx \succ Fg(n)$ in the following way:

$$\dfrac{\dfrac{\dfrac{}{(\forall x)Fx \succ (\forall x)Fx}\,[Id]}{(\forall x)Fx \succ Fn}\,[\forall Df]}{(\forall x)Fx \succ Fg(n)}\,[Spec^{n}_{g(n)}]$$

[8] The first paper in which a rule of this form is explicitly considered in a Gentzen system is Arnon Avron's 1993 paper "Gentzen-type systems, resolution and tableaux" [2].

Specification differs from the other sequent rules we have seen. It is not a local rule in a derivation introducing a single formula, but a global rule modifying the entire sequent.[9] In the classical sequent calculus it is an *admissible* rule, rather than a primitive rule, because any step in the derivation in which ends in a sequent involving a general term could be converted into a step in which that term is replaced by a more specific one. The rules [$\forall Df$] and [$\exists Df$] do not have this feature, when read from bottom to top.

Now consider what happens when we attempt to convert [$\forall Df$] into Left and Right rules using the technique of the previous section. The [$\forall R$] rule is the introduction half of [$\forall Df$]. For [$\forall L$] we can reason as follows:

$$\dfrac{\dfrac{\dfrac{}{(\forall x)A(x) \succ (\forall x)A(x)}\,[Id]}{(\forall x)A(x) \succ A(n)}\,[\forall Df] \qquad \Gamma, A(n) \succ \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[Cut]$$

Notice that the resulting derived rule

$$\dfrac{\Gamma, A(n) \succ \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[\forall L\text{: for names}]$$

no longer requires the side condition to the effect that n is absent from Γ or Δ. The Γ and Δ in this derivation are arbitrary, and do not pass through the inference step [$\forall Df$], where the side condition is in force. However, this algorithm does not give us the more general rule of the form

$$\dfrac{\Gamma, A(t) \succ \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[\forall L]$$

where t is an arbitrary singular term (not necessarily a deductively general one). To justify *this* form of the [$\forall L$] rule we must apply the algorithm not to an application of [$\forall Df$] to an identity sequent $(\forall x)A(x) \succ (\forall x)A(x)$, but to the result of *specifying* the result:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\forall x)A(x) \succ (\forall x)A(x)}\,[Id]}{(\forall x)A(x) \succ A(n)}\,[\forall Df]}{(\forall x)A(x) \succ A(t)}\,[Spec^{n}_{t}] \qquad \Gamma, A(t) \succ \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[Cut]$$

[9] So, it is not easily understood as corresponding to a natural deduction rule in which a proof is modified either at a premise or a conclusion position, but is rather understood as a global transformation of a proof. We transform a proof involving the deductively general term n into a proof involving the more specific term t.

which then gives us the fully general left rule $[\forall L]$. The same holds for the rules for the existential quantifier. Using $[\exists Df]$ and *Specification*, we can motivate the standard existential quantifier Right rule:

$$\frac{\Gamma \succ A(t), \Delta}{\Gamma \succ (\exists x)A(x), \Delta}\ [\exists R]$$

So, when we keep in mind the difference between deductively general terms and terms with more restricted behaviour (such as function terms), we can see that the standard Left and Right rules for the quantifiers can be understood in terms of their defining rules. In particular, the defining rule for a quantifier repends on the presence of deductively general terms, and the behaviour of these general terms—and their interaction with *Identity*, *Cut* and *Specification* generates the special asymmetric behaviour of these rules. One rule (the right rule for the universal quantifier; the left rule for the existential quantifier) invokes side conditions. The other rule does not.

3 QUANTIFIERS AND NON-DENOTING TERMS

However, the are reasons why we might want to reject these classical quantification rules. In particular, we may want to assert $(\forall x)Fx$ while denying Ft for some terms t in the language. For example, you might take it that according to the theory of real numbers we have $(\forall x)(x < 0 \vee x = 0 \vee x > 0)$—everything can be compared with 0—without concluding that $(\frac{1}{0} < 0 \vee \frac{1}{0} = 0 \vee \frac{1}{0} > 0)$—we don't want to claim that $\frac{1}{0}$ is comparable with 0—rather, this seems like something to be denied. One way to avoid concluding $(\frac{1}{0} < 0 \vee \frac{1}{0} = 0 \vee \frac{1}{0} > 0)$ is to banish items like "$\frac{1}{0}$" from the language. Perhaps we might banish the term on the grounds that, according to the theory of real numbers, division is not a total function. There is no such number as $\frac{1}{0}$ so $(\frac{1}{0} < 0 \vee \frac{1}{0} = 0 \vee \frac{1}{0} > 0)$ is not an instance of $(\forall x)(x < 0 \vee x = 0 \vee x > 0)$.

However, this is not, as a matter of fact, how everyday mathematical discourse regiments the language. The freedom of allowing singular terms such as $\frac{x}{y}$ and $\lim_{x\to\infty} f(x)$ which may occasionally fail to take values is a flexibility that is very useful for mathematical practice. It would be needlessly complicated to banish all such terms from our vocabulary. Rather, it seems that we may allow undefined terms in the vocabulary, provided that we are careful about the behaviour of those terms.

There are many ways to admit non-denoting terms to the vocabulary without doing too much violence to the norms of classical logic. One such attempt that is very congenial to the proof-theoretical framework we're exploring is due to Solomon Feferman, in his paper "Definedness" [5]. In that paper, Feferman provides a straightforward Hilbert-style proof system and Tarskian semantics for a logic extending classical first order predicate logic allowing for undefined singular terms. In the rest of this paper, I will show how considerations concerning generality and defining rules can independently motivate that logic.

If we are to admit undefined terms to the vocabulary, we need to be careful with regard to our rules for the quantifiers. No longer is the rule [∀Df] appropriate if the name n is allowed to be undefined—or rather, if n is deductively general, and an undefined term can be found in the same category as n. For otherwise, we could derive

$$(\forall x)(x < 0 \vee x = 0 \vee x > 0) \succ n < 0 \vee n = 0 \vee n > 0$$

and then, specify to the undefined term $\frac{1}{0}$. So, something has to give, if we are to accept $(\forall x)(x < 0 \vee x = 0 \vee x > 0)$ and reject $(\frac{1}{0} < 0 \vee \frac{1}{0} = 0 \vee \frac{1}{0} > 0)$. One option would be to deny that $\frac{1}{0}$ is in the same syntactic category as the deductively general term n. The name n is general, but it is not *so* general as to encompass the behaviour of non-denoting singular terms. This seems against the spirit of the enterprise, where we allow terms to be undefined, and where it is a matter of the *theory* (or the different views of the participants in a dialogue) as to whether terms are defined or not. That seems to be a matter of semantics, and not of syntactic category alone. We would very much like to allow the use of logical techniques in a discussion where we have participants who disagree not only about what is the case, but also disagree about what there is—in a shared vocabulary with an agreed upon syntactic regimentation.[10]

So, if we allow for assertion and denial of sentences, a natural generalisation is to allow for sequents to register *pro* and *con* attitudes toward *terms*, too. The guiding idea is that to rule a term *in* is to take it to denote, to allow it as an appropriate term to substitute into a quantifier. To rule a term *out* is to keep it in the vocabulary (mathematicians do not reject $\frac{1}{0}$ as syntactically ill-formed) but to not take it as a suitable substitution in generalisations. So, our sequents will not feature sentences alone onthe left and the right, but also terms. A sequent is now a pair $\Gamma \succ \Delta$ where each of Γ and Δ can contain terms as well as sentences.

How are we to reflect this change in the rules for quantifiers? The natural change to the defining rules for the universal and existential quantifier is to require in the premises the condition that the term n is *defined*.

$$\frac{\Gamma, n \succ A(n), \Delta}{\Gamma \succ (\forall x)A(x), \Delta}\ [\forall Df] \qquad \frac{\Gamma, n, A(n) \succ \Delta}{\Gamma, (\exists x)A(x) \succ \Delta}\ [\exists Df]$$

The upshot of denying $(\forall x)A(x)$ is to be prepared to deny $A(n)$ for a new name *which you take to be in order*. To assert $(\exists x)A(x)$ is to be prepared to grant $A(n)$ for a new name, again, taken to denote.

When we apply the algorithm to use *Identity*, *Cut* and *Specification* to generate

[10] This is not to say that fruitful and rational discussion *requires* agreement on the syntax of the shared language, just that some such discussion is possible with agreement on grammar without thereby requiring agreement on whether or not a singular term denotes.

the remaining Left/Right rule for each quantifier, the following derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\forall x)A(x) \succ (\forall x)A(x)}\,[Id]}{(\forall x)A(x), n \succ A(n)}\,[\forall Df]}{(\forall x)A(x), t \succ A(t)}\,[Spec^n_t] \qquad \Gamma, A(t) \succ \Delta}{\Gamma, (\forall x)A(x), t \succ \Delta}\,[Cut] \qquad \Gamma \succ t, \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[t\text{-}Cut]$$

generates the $[\forall L]$ rule:

$$\dfrac{\Gamma, A(t) \succ \Delta \quad \Gamma \succ t, \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[\forall L]$$

Similarly, the the derivation for the existential quantifier

$$\dfrac{\Gamma, t \succ \Delta \qquad \dfrac{\Gamma \succ A(t), \Delta \quad \dfrac{\dfrac{\dfrac{\dfrac{}{(\exists x)A(x) \succ (\exists x)A(x)}\,[Id]}{(\exists x)A(x) \succ (\exists x)A(x)}\,[\exists Df]}{A(n), n \succ (\exists x)A(x)}\,[Spec^n_t]}{A(t), t \succ (\exists x)A(x)}}{\Gamma, t \succ (\exists x)A(x), \Delta}\,[Cut]}{\Gamma \succ (\exists x)A(x), \Delta}\,[t\text{-}Cut]$$

generates this $[\exists R]$ rule:

$$\dfrac{\Gamma \succ t, \Delta \quad \Gamma \succ A(t), \Delta}{\Gamma \succ (\exists x)A(x), \Delta}\,[\exists R]$$

The second *Cut* step in the justification for the $[\forall L]$ rule was a t-*Cut*, a *Cut* on a term. This is form of the *Cut* rule

$$\dfrac{\Gamma \succ t, \Delta \quad \Gamma, t \succ \Delta}{\Gamma \succ \Delta}\,[t\text{-}Cut]$$

is just as motivated as a *Cut* on sentences. (If there is a clash involved ruling t *in* and in ruling t *out* in a context, there is a clash in the underlying context.)

To complete the motivation of our logic with non-denoting terms we need to consider the interaction between denotation, predication and function terms, and to consider the behaviour of an 'existence predicate' which makes explicit in assertion and denial the term judgements involved in ruling in and ruling out terms. The latter is straightforward. Here is a defining rule for the existence predicate:

$$\dfrac{\Gamma, t \succ \Delta}{\Gamma, t\!\downarrow \succ \Delta}\,[\downarrow Df]$$

'$\downarrow$' is syntactically a predicate, and it makes explicit the judgement that the term t denotes as a *sentence*. To rule $\frac{1}{0}$ as undefined is to deny $\frac{1}{0}\downarrow$—to assert its negation $\neg\frac{1}{0}\downarrow$. Generating the *Right* rule for $\downarrow$ is straightforward:

$$\dfrac{\Gamma \succ t, \Delta \quad \dfrac{\dfrac{}{t\downarrow \succ t\downarrow}\,[Id]}{t \succ t\downarrow}\,[\downarrow Df]}{\Gamma \succ t\downarrow, \Delta}\,[t\text{-}Cut]$$

The right rule replaces a term judgement t in the right hand side of a sequent with the corresponding sentence $t\downarrow$, as expected.

For function terms, a natural constraint is that an n-place function f is defined on inputs $t_1, \ldots, t_n$ only when those inputs are defined.[11] This motivates the following interaction rule for term judgements and function application:

$$\frac{t_i, \Gamma \succ \Delta}{f(t_1, \ldots, t_n), \Gamma \succ \Delta}\,[fL]$$

Feferman's free logic does makes the same choice for *predication*. Given a primitive n-ary predicate F, it can be *truly applied* to the terms $t_1, \ldots, t_n$ only when those terms are defined.

$$\frac{t_i, \Gamma \succ \Delta}{Ft_1 \cdots t_n, \Gamma \succ \Delta}\,[FL]$$

This seems very natural in the mathematical case ($\frac{1}{0}$ is not even, and neither is $\frac{1}{0}$ prime), so we will adopt the rule here. Note that adopting such a rule involves drawing a distinction between primitive and complex predicates. While $\frac{1}{0}$ is not even, the complex predicate 'not even' is *truly applied* to the term $\frac{1}{0}$. This rule, while satisfied by simple predicates F is not necessarily satisfied by other complex predicates—and of course, it cannot be satisfied by the negation of the definedness predicate $\downarrow$. That *only* truly applies to non-denoting terms by design.[12]

With this motivation behind us, let's see where we have arrived, and examine this proof system in some detail.

4 DERIVATIONS AND SYSTEMS

To be relatively precise, let's assume we work with a language consisting of a supply of NAMES ($m, n, n_1, n_2, \ldots$), and function symbols (f, g, etc.), and PREDICATES

[11] So, on this view, $\frac{1}{0} - \frac{1}{0}$ is undefined, even if the function $\lambda x.(x - x)$ is the constant zero function that is everywhere defined. To use the distinction from computer science, we adopt a call-by-value semantics rather than a call-by-name semantics for function evaluation [14]. Inputs to functions are evaluated *before* the evaluation of the function.

[12] In a sequel to this paper, I will discuss this issue further, when examining generality and *predicates*. It is enough to leave that intersection in the highway of options for later, and to follow along with Feferman in this choice for free logic and predication.

(F, G, L, R, . . .). We include a special one-place predicate '↓', to be written *postfix*. A TERM in the language is either a name or a function symbol applied to an appropriate number of terms.

We also have an infinite supply of VARIABLES ($x, y, z, x_1, x_2, \ldots$) to feature in our quantifiers. Formulas are combined with the usual CONNECTIVES $\wedge$, $\vee$, $\supset$ and $\neg$. Sentences do not contain any free variables. To apply a quantifier $(\exists x)$ to a sentence $A(n)$ with some number of instances of the name n designated, replace the name n by variable x and prefix the result with the quantifier.

We use the capital Greek letters Γ and Δ variously for sets and multisets of sentences and terms. Two multisets are identical if they have the same members the same number of times, and two sets are identical if they merely have the same members. We write these multisets and sets in *list* notation without any parentheses since the only members are sentences or terms, and we use a comma to indicate adding a member to a set or multiset. (So, if t is a term and Γ is a multiset, then t, Γ is another multiset that contains everything to the same degree that Γ does, except for t, which it contains one *more* time than Γ does.)

In our sequent calculus, sequents are pairs of multisets, $\Gamma \succ \Delta$ consisting of sentences and terms. We call Γ the LEFT HAND SIDE (LHS) of the sequent and Δ its right hand side (RHS). The sequent calculus defines a class of DERIVATIONS of sequents—trees of sequents constructed using the following rules. The *leaves* of derivations are *Identity* sequents of the following shapes:

$$\Gamma, A \succ A, \Delta \text{ [s-}Id\text{]} \qquad \Gamma, t \succ t, \Delta \text{ [t-}Id\text{]}$$

These identity sequents incorporate the weakening rule. The presence of irrelevant extra formulas in the sequent is allowed.

The *Identity* rules are *structural* rules. A structural rule pays no heed to the internal structure of sentences or terms. Another structural rule is the rule of *Contraction*, which comes in four flavours: two *left* rules, and two *right* rules, two *sentence* rules and two *term* rules.

$$\frac{\Gamma, A, A \succ \Delta}{\Gamma, A \succ \Delta} \text{ [s-}WL\text{]} \qquad \frac{\Gamma \succ A, A, \Delta}{\Gamma \succ A, \Delta} \text{ [s-}WR\text{]} \qquad \frac{\Gamma, t, t \succ \Delta}{\Gamma, t \succ \Delta} \text{ [t-}WL\text{]} \qquad \frac{\Gamma \succ t, t, \Delta}{\Gamma \succ, \Delta} \text{ [t-}WR\text{]}$$

The contraction rules ensure that repeats of formulas or terms have no effect on the consequence relation.[13] Our next pair of structural rules are the *Cut* rules, again, one for a cut on a *sentence* and the other for a cut on a *term*.

$$\frac{\Gamma \succ A, \Delta \quad \Gamma, A \succ \Delta}{\Gamma \succ \Delta} \text{ [s-}Cut\text{]} \qquad \frac{\Gamma \succ t, \Delta \quad \Gamma, t \succ \Delta}{\Gamma \succ \Delta} \text{ [t-}Cut\text{]}$$

[13] You may wonder why we work with multisets of formulas, which allow for repeats, while sets of formulas do not. The difference is important for when we consider derivations as representing the structure of a *proof*. If a sequent $A, B \succ A \wedge B$ makes use of two premises, one used to justify the first conjunct of the conclusion, and the other to justify the second, then the proof ould have still had that shape had A and B been the *same formula*. A proof step from A and A to $A \wedge A$ is, in some sense, a different proof from a proof step from A to $A \wedge A$, even though they are equally valid.

Finally, we have the non-local rule of *specification*, which allows for the global replacement of a deductively general term n (a *name*) by a term t in the entire sequent.

$$\frac{\Gamma \succ \Delta}{\Gamma[n := t] \succ \Gamma[n := t]}\ [Spec^n_t]$$

That rounds out the *structural* rules of the calculus. To those we add the rules for function application and predication. These are not structural rules—they dictate the interaction between definedness of terms and function terms and predications, and so, they depend on the particular form of the sentences displayed in the rules.

$$\frac{t_i, \Gamma \succ \Delta}{f(t_1, \ldots, t_n), \Gamma \succ \Delta}\ [fL] \qquad \frac{t_i, \Gamma \succ \Delta}{Ft_1 \cdots t_n, \Gamma \succ \Delta}\ [FL]$$

Yet,these are not *defining* rules for any particular vocabulary. They are certainly not defining rules for the function terms or predicates, because they are the same shape rule for each predicate and function symbol of a given arity.

We have one DEFINING RULE for each logical concept in the vocabulary. Here are the defining rules for the propositional connectives:

$$\frac{\Gamma, A, B \succ \Delta}{\Gamma, A \wedge B \succ \Delta}\ [\wedge Df] \quad \frac{\Gamma \succ A, B, \Delta}{\Gamma \succ A \vee B, \Delta}\ [\vee Df] \quad \frac{\Gamma, A \succ B, \Delta}{\Gamma \succ A \supset B, \Delta}\ [\supset Df] \quad \frac{\Gamma \succ A, \Delta}{\Gamma, \neg A \succ \Delta}\ [\neg Df]$$

And here are the defining rules for the quantifiers and the definedness predicate:

$$\frac{\Gamma, n \succ A(n), \Delta}{\Gamma \succ (\forall x)A(x), \Delta}\ [\forall Df] \qquad \frac{\Gamma, n, A(n) \succ \Delta}{\Gamma, (\exists x)A(x) \succ \Delta}\ [\exists Df] \qquad \frac{\Gamma, t \succ \Delta}{\Gamma, t\downarrow \succ \Delta}\ [\downarrow Df]$$

As usual, the quantifier rules have the usual side condition: the name n (a deductively general term) is not present in the conclusion of the rule. Let's call the system DL[*Df, Cut, Spec*] the proof system for definedness logic with *Defining Rules*, the *Cut* rules and the *Specification Rule*.

A DERIVATION of a sequent $\Gamma \succ \Delta$ in DL[*Df, Cut, Spec*] is a tree of sequents, starting with *Identity* sequents at the leaves, each step of which is an instance of one of the rules, and which ends at the root in $\Gamma \succ \Delta$. Here is an example derivation in the proof system, of $(\forall x)A(x), t\downarrow \succ A(t)$.

$$\dfrac{\dfrac{\dfrac{(\forall x)A(x) \succ (\forall x)A(x)}{(\forall x)A(x), n \succ A(n)}\ [\forall Df]}{(\forall x)A(x), t \succ A(t)}\ [Spec^n_t]}{(\forall x)A(x), t\downarrow \succ A(t)}\ [\downarrow Df]$$

Here is a longer derivation, of the sequent $(\forall x)(Fx \supset Gx) \succ (\exists x)Fx \supset (\exists x)Gx$.

$$\dfrac{\dfrac{(\forall x)(Fx \supset Gx) \succ (\forall x)(Fx \supset Gx)}{(\forall x)(Fx \supset Gx), n \succ Fn \supset Gn}\,[\forall Df] \quad \dfrac{\dfrac{Fn \supset Gn \succ Fn \supset Gn}{Fn \supset Gn, Fn \succ Gn}\,[\supset Df] \quad \dfrac{(\exists x)Gx \succ (\exists x)Gx}{n, Gn \succ (\exists x)Gx}\,[\exists Df]}{Fn \supset Gn, n, Fn \succ (\exists x)Gx}\,[s\text{-}Cut]}{\dfrac{(\forall x)(Fx \supset Gx), n, Fn \succ (\exists x)Gx}{(\forall x)(Fx \supset Gx), (\exists x)Fx \succ (\exists x)Gx}\,[\exists Df]}\,[s\text{-}Cut]$$
$$\dfrac{(\forall x)(Fx \supset Gx), (\exists x)Fx \succ (\exists x)Gx}{(\forall x)(Fx \supset Gx) \succ (\exists x)Fx \supset (\exists x)Gx}\,[\supset Df]$$

(In fact, this derivation is not formally correct as it is written. In order for the derivation to fit on the page, I have left out some irrelevant side formulas from instances of the *Cut* rule. Take the first *s-Cut* step, concluding in $Fn \supset Gn, n, Fn \succ (\exists x)Gx$. The *Cut* step is a cut on the formula Gn, so the two premises hould be the sequents $Fn \supset Gn, n, Fn \succ Gn, (\exists x)Gx$ and $Fn \supset Gn, n, Fn, Gn \succ (\exists x)Gx$. Instead, the left premise uses only the underlined formulas in $\underline{Fn \supset Gn}, \underline{n}, \underline{Fn} \succ \underline{Gn}, (\exists x)Gx$, leaving out the final formula (which is not used in the derivation of that sequent) and the second premise makes use only of $Fn \supset Gn, \underline{n}, Fn, \underline{Gn} \succ \underline{(\exists x)Gx}$, leaving the rest aside. The full derivation, according to the formal definitions of the rules, would be much wider, and we have left out unused formulas for the sake of space.)

As we have seen in the first two sections of this paper, and *Specification* can be eliminated as a rule if we are willing to generalise the *Defining Rules*, and the *Defining Rules* can be traded in for the traditional pairs of Left/Right rules of a sequent calculus. We will end this section precisely stating and proving these two facts. First, the elimination of *Specification*. The *Generalised Defining Rules* for the quantifiers are given as follows:

$$\dfrac{\Gamma, n \succ A(n), \Delta}{\Gamma \succ (\forall x)A(x), \Delta}\,[\forall Df\Downarrow] \qquad \dfrac{\Gamma \succ (\forall x)A(x), \Delta}{\Gamma, t \succ A(t), \Delta}\,[\forall Df\Uparrow]$$

$$\dfrac{\Gamma, n, A(n) \succ \Delta}{\Gamma, (\exists x)A(x) \succ \Delta}\,[\exists Df\Downarrow] \qquad \dfrac{\Gamma, (\exists x)A(x) \succ \Delta}{\Gamma, t, A(t) \succ \Delta}\,[\exists Df\Uparrow]$$

The $\Downarrow$ parts of these rules are one half of each of the original defining rules for the quantifiers, and they retain the side condition to the effect that the name n does not appear in the concluding sequent. The $\Uparrow$ parts of the rules are generalisations of the other halfs, allowing an implicit specification from the deductively general name n to the more specific term t. Let's call the system DL[*GDf, Cut*] the proof system for definedness logic with the *Generalised Defining Rules*, and the *Cut* rules but without the *Specification Rule*.

FACT 1: *A derivation of a sequent* $\Gamma \succ \Delta$ *in* DL[*Df, Cut, Spec*] *can be systematically transformed into a derivation of that sequent in* DL[*GDf, Cut*]*, and vice versa.*

Proof: For the left-to-right direction, we argue as follows: The rule of specification is admissible in DL[*GDf, Cut*] in the following sense. Any derivation in DL[*GDf, Cut*] of a sequent $\Gamma \succ \Delta$ can be transformed into a DL[*GDf, Cut*] derivation of the sequent $\Gamma[n := t] \succ \Gamma[n := t]$. Consider each of the rules in DL[*GDf, Cut*]. They are all closed under specification, in the sense that if a rule ends in some sequent $\Gamma \succ \Delta$ from some premise sequents, another instance of that rule leads to the conclusion $\Gamma[n := t] \succ \Delta[n := t]$ in which each of the instances of n in that sequent are replaced by t. There are no rules in which a *name* occurs in a conclusion sequent where it could not have been everywhere replaced by an arbitrary *term*. (This is why we replaced the direction of the defining rules for the quantifier in which the name occurred in the concluding sequent. They are the only rules in the system to not have this propertty.) So, given any derivation in DL[*Df, Cut, Spec*], replace each appeal to the *Specification* rule by a transformation of the derivation of the premise of that rule into a DL[*GDf, Cut*] derivation of the conclusion. The end result of this process is a derivation of the concluding sequent in DL[*GDf, Cut*].

Conversely, any DL[*GDf, Cut*] derivation can be transformed into a DL[*Df, Cut, Spec*] derivation by replacing each instance of a $[\forall Df\Uparrow]$ or $[\exists Df\Uparrow]$ step by a $[\forall Df]$ or $[\exists Df]$ step composed with the appropriate instance of $[Spec^n_t]$ to convert the name in the conclusion of the *Df* step inot the term required for the conclusion of the corresponding *GDf* step. The result is a derivation of our end sequent $\Gamma \succ \Delta$. ∎

The process for replacing *Defining Rules* by *Left* and *Right* rules is also straightforward. As we have seen, the defining rules for the connectives, quantifiers and existence predicate can be replaced by the following Left/Right rules. The system DL[*LR, Cut*] consists of the structural rules of *Identity*, *Contraction* and *Cut* (so we leave out *Specification*), the predicate and function term rules, and these *Left* and *Right* rules for the connectives, quantifiers and definedness predicate.

$$\frac{\Gamma, A, B \succ \Delta}{\Gamma, A \wedge B \succ \Delta}\,[\wedge L] \qquad \frac{\Gamma \succ A, \Delta \quad \Gamma \succ B, \Delta}{\Gamma \succ A \wedge B, \Delta}\,[\wedge R]$$

$$\frac{\Gamma, A \succ \Delta \quad \Gamma, B \succ \Delta}{\Gamma, A \vee B \succ \Delta}\,[\vee L] \qquad \frac{\Gamma \succ A, B, \Delta}{\Gamma \succ A \vee B, \Delta}\,[\vee R]$$

$$\frac{\Gamma \succ A, \Delta \quad \Gamma, B \succ \Delta}{\Gamma, A \supset B \succ \Delta}\,[\supset L] \qquad \frac{\Gamma, A \succ B, \Delta}{\Gamma \succ A \supset B, \Delta}\,[\supset R]$$

$$\frac{\Gamma \succ A, \Delta}{\Gamma, \neg A, \succ \Delta}\,[\neg L] \qquad \frac{\Gamma, A \succ \Delta}{\Gamma \succ \neg A, \Delta}\,[\neg R]$$

$$\frac{\Gamma, A(t) \succ \Delta \quad \Gamma \succ t, \Delta}{\Gamma, (\forall x)A(x) \succ \Delta}\,[\forall L] \qquad \frac{\Gamma, n \succ A(n), \Delta}{\Gamma \succ (\forall x)A(x), \Delta}\,[\forall R]$$

$$\frac{\Gamma, n, A(n) \succ \Delta}{\Gamma, (\exists x)A(x) \succ \Delta}\,[\exists L] \qquad \frac{\Gamma \succ t, \Delta \quad \Gamma \succ A(t), \Delta}{\Gamma \succ (\exists x)A(x), \Delta}\,[\exists R]$$

$$\frac{\Gamma, t \succ \Delta}{\Gamma, t\downarrow \succ \Delta}\,[\downarrow L] \qquad \frac{\Gamma \succ t, \Delta}{\Gamma \succ t\downarrow, \Delta}\,[\downarrow R]$$

FACT 2: *A derivation of a sequent* $\Gamma \succ \Delta$ *in* DL[*LR, Cut*] *can be systematically transformed into a derivation of that sequent in* DL[*GDf, Cut*]*, and vice versa.*

Proof: The examples of the previous section show how each Left/Right rule can be constructed out of the defining rules together with *Identity* and *Cut*, so we may take any derivation in DL[*LR, Cut*] and replace each appeal to a Left or Right rule with the corresponding appeal to the Defining rule for the concept involved, composed with appropriate *Cut* and *Identity* steps.

We have not yet shown the converse. We wish to show how appeals to defining rules can be reconstructed in DL[*LR, Cut*]. For each step of a defining rule in which the defined concept is introduced in the conclusion of the rule (the *downward* half of the rule), that part of the rule is either a Left or Right rule in DL[*LR, Cut*]. Our work is to reconstruct the *upward* half. Here are some examples, for $\supset$, $\forall$ and $\downarrow$. The others are natural generalisations of this technique.

$$\dfrac{\dfrac{A \succ A \quad B \succ B}{A, A \supset B \succ B}\,[\supset L] \qquad \Gamma \succ A \supset B, \Delta}{\Gamma, A \succ B, \Delta}\,[s\text{-}Cut]$$

$$\dfrac{\dfrac{A(t), t \succ A(t) \quad t \succ t, A(t)}{(\forall x)A(x), t \succ A(t)}\,[\forall L] \qquad \Gamma \succ (\forall x)A(x), \Delta}{\Gamma, t \succ A(t), \Delta}\,[s\text{-}Cut]$$

$$\dfrac{\dfrac{t \succ t}{t \succ t\downarrow}\,[\downarrow R] \qquad \Gamma, t\downarrow \succ \Delta}{\Gamma, t \succ \Delta}\,[s\text{-}Cut]$$

■

So, in this section we have defined three different proof systems for DL: DL[*Df, Cut, Spec*], with *Defining Rules* and *Specification*, DL[*GDf, Cut*], with *Generalised Defining Rules*, where specification is no longer a primitive rule but is folded into the remaining rules, and DL[*LR, Cut*], which replaces the defining rules with traditional Left and Right rules of a Gentzen-style sequent calculus. These proof systems are motivated by normative pragmatic considerations—derivable sequents record norms governing assertion and denial in a language which allows for singular terms with reference failure.

Each of these three proof systems makes use of *Cut* rules, both the traditional s-*Cut* for cuts on sentences, and the novel t-*Cut* for cuts on terms. In the next two sections we will move to show that DL[*LR,Cut*] is equivalent to DL[*LR*], that any

sequent derivable with the *Cut* rules is also derivable without them. To demonstrate this, we will more closely analyse those sequents which are *not* derivable in these proof systems, and explore the connection between underivable sequents and models [10].

5 POSITIONS AND REFINEMENT

If a sequent $\Gamma \succ \Delta$ cannot be derived, then, as far as the norms of our logic are concerned, there is no clash involved in asserting everything in Γ and denying something in Δ. If we think of the logic as informing and inscribing a field of play in which various assertions and denials are made and various terms are ruled in or ruled out, the underivable sequents $\Gamma \succ \Delta$ are different *positions* in that field of play.

To be precise, let us take a POSITION (relative to a proof system) to be a pair $[\Gamma : \Delta]$ of *sets* Γ and Δ where every $\Gamma' \succ \Delta'$ is underivable in that proof system, for any finite multisets Γ' and Δ' where each element of Γ' is in Γ and every element of Δ' is in Δ.

In this definition the components of a position differ from the LHS and RHS of a sequent in two ways. First, we do not care to keep track of the repetitions of sentences or terms in Γ and Δ, so we take them to be *sets* and not *multisets*. Furthermore, we allow Γ and Δ to be *infinite*, while the LHS and RHS in a sequent in our proof systems are always *finite*. If $[\Gamma : \Delta]$ is a position, then there is no clash involved in selecting any assertions from Γ and any denials from Δ, however many we take.

Let us say that the position $[\Gamma_2 : \Delta_2]$ is a REFINEMENT of the position $[\Gamma_1 : \Delta_1]$ if and only if $\Gamma_1 \subseteq \Gamma_2$ and $\Delta_1 \subseteq \Delta_2$.

We will say that a position $[\Gamma : \Delta]$ is FINITE if and only if the sets Γ and Δ are finite.

Our focus will be on positions in the system DL[*LR*], and our aim will be, in this section and the next, to show that those sequents which can not be derived without the use of the *Cut* rules could not have been derived *with* those rules, either. We will show this by way of a model construction, showing how DL[*LR*] positions may be "filled out" or "idealised" into models which serve as a witness (a model witnessing a position will evaluate each formula in its LHS as true, each term in its LHS as defined, each formula in its RHS as false, and each term in its RHS as undefined). These models will turn out to be models of the system DL[*LR*,*Cut*] too—it will be straightforward to show that these models only witnesses to positions in DL[*LR*,*Cut*], so it will follow that DL[*LR*] positions are in fact DL[*LR*,*Cut*] positions, so any sequent derivable *with Cuts* is derivable without them. That is our target in this section and the next.

To start, let us focus on how positions may be progressively refined to make them more informative. First, let's consider a DL[*LR*] position in which $A \wedge B$ occurs

in the LHS: $[\Gamma, A \wedge B : \Delta]$. Suppose $[\Gamma, A \wedge B, A, B : \Delta]$ weren't a position. Then it would follow that for some $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ that there is a derivation of $\Gamma', A \wedge B, A, B \succ \Delta'$. It would follow that there is a derivation of $\Gamma', A \wedge B \succ \Delta'$ too, and that (contrary to hypothesis) $[\Gamma, A \wedge B : \Delta]$ is not a DL[*LR*] position. Here is why:

$$\dfrac{\dfrac{\Gamma', A \wedge B, A, B \succ \Delta'}{\Gamma', A \wedge B, A \wedge B \succ \Delta'}\,[\wedge L]}{\Gamma', A \wedge B \succ \Delta'}\,[s\text{-}WL]$$

So, if $[\Gamma, A \wedge B : \Delta]$ is a position, so is $[\Gamma, A \wedge B, A, B : \Delta]$. This is called a LEFT-CONJUNCTION REFINEMENT of $[\Gamma, A \wedge B : \Delta]$.

Now consider a position in which $A \wedge B$ occurs in the RHS: $[\Gamma : A \wedge B, \Delta]$. We can show that at least one of $[\Gamma : A, A \wedge B, \Delta]$ and $[\Gamma : B, A \wedge B, \Delta]$ are also positions, because if they fail to be positions, $[\Gamma : A \wedge B, \Delta]$ must fail to be a position too:

$$\dfrac{\dfrac{\Gamma' \succ A, A \wedge B, \Delta' \quad \Gamma' \succ B, A \wedge B, \Delta'}{\Gamma' \succ A \wedge B, A \wedge B, \Delta'}\,[\wedge R]}{\Gamma' \succ A \wedge B, \Delta'}\,[s\text{-}WR]$$

So, if $[\Gamma : A \wedge B, \Delta]$ is a position, then at least one of $[\Gamma : A, A \wedge B, \Delta]$ and $[\Gamma : B, A \wedge B, \Delta]$ is a position too. Those that are positions are called RIGHT-CONJUNCTION REFINEMENTS of $[\Gamma : A \wedge B, \Delta]$.

In the same way we can define refinements for *disjunctions*, *conditionals* and *negations* occuring in the LHS or RHS of positions. The complete table of refinements for propositional connectives is given in Figure 1.

POSITION	REFINEMENTS
$[\Gamma, A \wedge B : \Delta]$	$[\Gamma, A \wedge B, A, B : \Delta]$
$[\Gamma : A \wedge B, \Delta]$	At least one of $[\Gamma : A, A \wedge B, \Delta]$ and $[\Gamma : B, A \wedge B, \Delta]$
$[\Gamma, A \vee B : \Delta]$	At least one of $[\Gamma, A, A \vee B : \Delta]$ and $[\Gamma, B, A \vee B : \Delta]$
$[\Gamma : A \vee B, \Delta]$	$[\Gamma : A, B, A \vee B, \Delta]$
$[\Gamma, A \supset B : \Delta]$	At least one of $[\Gamma, A \supset B : A, \Delta]$ and $[\Gamma, B, A \supset B : \Delta]$
$[\Gamma : A \supset B, \Delta]$	$[\Gamma, A : B, A \vee B, \Delta]$
$[\Gamma, \neg A : \Delta]$	$[\Gamma, \neg A : A, \Delta]$
$[\Gamma : \neg A, \Delta]$	$[\Gamma, A : \neg A, \Delta]$

Figure 1: Refinements for propositional connectives

.

Propositional refinement relations fill out our positions to carry more information concerning the components of sentences in positions. The same will hold for the definedness predicate and the quantifiers.

If $[\Gamma, t\downarrow : \Delta]$ a position, then $[\Gamma, t\downarrow, t : \Delta]$ is a LEFT-DEFINEDNESS REFINEMENT of that position. If $[\Gamma : t\downarrow, \Delta]$ a position, then $[\Gamma : t, t\downarrow, \Delta]$ is a RIGHT-DEFINEDNESS REFINEMENT of that position. The left-definedness and right-definedness refinements of positions are themselves positions for the following reasons: A derivation that shows a refinement fails to be a position would show that the position of which it is a refinement were also not a position, contrary to hypothesis:

$$\dfrac{\dfrac{\Gamma', t\downarrow, t \succ \Delta'}{\Gamma', t\downarrow, t\downarrow \succ \Delta'}\,[\downarrow L]}{\Gamma', t\downarrow \succ \Delta'}\,[s\text{-}WL] \qquad \dfrac{\dfrac{\Gamma' \succ t, t\downarrow, \Delta'}{\Gamma' \succ t\downarrow, t\downarrow, \Delta'}\,[\downarrow R]}{\Gamma' \succ t\downarrow, \Delta'}\,[s\text{-}WR]$$

Similarly, we can define refinements for formulas featuring the quantifiers. Consider a position in which $(\forall x)A$ occurs in the LHS: $[\Gamma, (\forall x)A(x) : \Delta]$. Then one of $[\Gamma, (\forall x)A(x), A(t) : \Delta]$ and $[\Gamma, (\forall x)A(x) : t, \Delta]$ is a position, too, where t is any of the terms in the language. If these both failed to be a position, so would our starting position:

$$\dfrac{\dfrac{\Gamma', (\forall x)A(x), A(t) \succ \Delta' \quad \Gamma', (\forall x)A(x) \succ, t, \Delta'}{\Gamma', (\forall x)A(x), (\forall x)A(x) \succ \Delta'}\,[\forall L]}{\Gamma', (\forall x)A(x) \succ \Delta'}\,[s\text{-}WL]$$

For the RHS, consider the position $[\Gamma : (\forall x)A, \Delta]$. Then $[\Gamma, n : A(n), (\forall x)A(x), \Delta]$ is a position too, where n name not occuring in $[\Gamma : (\forall x)A(x), \Delta]$.

$$\dfrac{\dfrac{\Gamma', n \succ A(n), (\forall x)A(x), \Delta'}{\Gamma' \succ (\forall x)A(x), (\forall x)A(x), \Delta'}\,[\forall R]}{\Gamma' \succ (\forall x)A(x), \Delta'}\,[s\text{-}WR]$$

Dual reasoning applies to the existential quantifier rules. The table in Figure 2 presents refinements for quantifiers and the definedness predicate.

We aim to find positions which are closed under these particular refinement relations, for these will be our guide to constructing a *model*. Consider refinement for conjunctions. If a conjunction is present in the LHS of such a position, so are its conjuncts. If a conjunction is in the RHS of such a position, so is one of its conjuncts (at least). Being on the LHS does a good job a proxy for *truth* and being on the RHS does the same for *falsity*, at least as far as conjuction goes. For the definedness predicate, if the position places $t\downarrow$ on the LHS, if it is closed under the definedness refinement conditions, t is also on the LHS. If $t\downarrow$ is in the RHS, then so is t. So, if a term is on the LHS, according to that position, it is defined. If it is in the RHS, then according to that position it is undefined. For the quantifiers, if $(\forall x)A(x)$ is in the LHS of a position, and it is closed under quantifier refinement for each term in its own vocabulary, then for every such term t, either t is in the RHS (so, according to that position, t is undefined) or $A(t)$ is in the LHS (so, $A(t)$ is taken to be true).

POSITION	REFINEMENTS
$[\Gamma, (\forall x)A(x) : \Delta]$	At least one of $[\Gamma, (\forall x)A(x), A(t) : \Delta]$, $[\Gamma, (\forall x)A(x) : t, \Delta]$ for each term t in $[\Gamma, (\forall x)A(x) : \Delta]$.
$[\Gamma : (\forall x)A(x), \Delta]$	$[\Gamma, n : A(n), (\forall x)A(x), \Delta]$, where n is new.
$[\Gamma, (\exists x)A(x) : \Delta]$	$[\Gamma, (\exists x)A(x), A(n), n : \Delta]$, where n is new.
$[\Gamma : (\exists x)A(x), \Delta]$	At least one of $[\Gamma : A(t), (\exists x)A(x), \Delta]$, $[\Gamma : t, (\exists x)A(x), \Delta]$ for each term t in $[\Gamma : (\exists x)A(x), \Delta]$.
$[\Gamma, t\downarrow : \Delta]$	$[\Gamma, t\downarrow, t : A, \Delta]$
$[\Gamma : t\downarrow, \Delta]$	$[\Gamma, A : t, t\downarrow, \Delta]$

Figure 2: Refinements for quantifiers and the definedness predicate

.

Similarly, if $(\forall x)A(x)$ is in the RHS of a position, then we have added a name n to the vocabulary of the position such that n occurs in the LHS and $A(n)$ in the RHS. In other words, if the position takes $(\forall x)A(x)$ to be false, then we have equipped the vocabulary with some name n where according to the position n denotes, and $A(n)$ is false.

In other words, positions closed under the refinement conditions do a good job of respecting natural truth conditions, at least for conjunction, definedness and the universal quantifier. But that is not *quite* enough for us to define models in which the entire logic is respected. For that, we need to consider the interaction between definedness, predication and function application. We would like to find positions which fully respect the conditions on predication and function application. If $Ft_1 \cdots t_n$ is true, then each t_i is defined. If $f(t_1, \ldots, t_n)$ is defined, then each t_i is defined. So, we will refine positions for predication and function

POSITION	REFINEMENTS
$[\Gamma, Ft_1 \cdots t_n : \Delta]$	$[\Gamma, Ft_1 \cdots t_n, t_1, \ldots, t_n : \Delta]$
$[\Gamma, f(t_1, \ldots, t_n) : \Delta]$	$[\Gamma, f(t_1, \ldots, t_n), t_1, \ldots, t_n : \Delta]$

Figure 3: Refinements for predication and function application

.

application in the following way. If a position takes a predication to be true, we will add the terms of that predication to the LHS. If a position features a function application in the LHS, we add the terms of that function application to the LHStoo. The result in either case is a position, as the following derivations show, first for

predication:

$$\dfrac{\dfrac{\dfrac{\Gamma', Ft_1 \cdots t_n, t_1, \ldots, t_n \succ \Delta'}{\Gamma', Ft_1 \cdots t_n, Ft_1 \cdots t_n, t_2, \ldots, t_n \succ \Delta'}\,[FL]}{\Gamma', Ft_1 \cdots t_n, t_2, \ldots, t_n \succ \Delta'}\,[s\text{-}WL]}{\vdots}$$

$$\dfrac{\dfrac{\dfrac{\Gamma', Ft_1 \cdots t_n, Ft_1 \cdots t_n, t_n \succ \Delta'}{\Gamma', Ft_1 \cdots t_n, t_n \succ \Delta'}\,[s\text{-}WL]}{\Gamma', Ft_1 \cdots t_n, Ft_1 \cdots t_n \succ \Delta'}\,[FL]}{\Gamma', Ft_1 \cdots t_n \succ \Delta'}\,[s\text{-}WL]$$

and then for function application:

$$\dfrac{\dfrac{\dfrac{\Gamma', f(t_1, \ldots, t_n), t_1, \ldots, t_n \succ \Delta'}{\Gamma', f(t_1, \ldots, t_n), f(t_1, \ldots, t_n), t_2, \ldots, t_n \succ \Delta'}\,[fL]}{\Gamma', f(t_1, \ldots, t_n), t_2, \ldots, t_n \succ \Delta'}\,[t\text{-}WL]}{\vdots}$$

$$\dfrac{\dfrac{\dfrac{\Gamma', f(t_1, \ldots, t_n), f(t_1, \ldots, t_n), t_n \succ \Delta'}{\Gamma', f(t_1, \ldots, t_n), t_n \succ \Delta'}\,[t\text{-}WL]}{\Gamma', f(t_1, \ldots, t_n), f(t_1, \ldots, t_n) \succ \Delta'}\,[fL]}{\Gamma', f(t_1, \ldots, t_n) \succ \Delta'}\,[t\text{-}WL]$$

This completes the definition of the refinement relations for the connectives, the quantifiers, the definedness predicate, and for primitive predication and function application. Now we will use this to construct a position closed under each of these relations in the appropriate fashion.[14]

A set $\mathcal{D}$ of positions is said to be DIRECTED if it is (1) closed downward under refinement, in the sense that if $[\Gamma : \Delta]$ is refined by some $[\Gamma' : \Delta']$ in $\mathcal{D}$, then $[\Gamma : \Delta]$ is in $\mathcal{D}$ too; and (2) $\mathcal{D}$ contains upper bounds, in the sense that if $[\Gamma_1 : \Delta_1]$ and $[\Gamma_2 : \Delta_2]$ are in $\mathcal{D}$ then there is some position refining both that is also in $\mathcal{D}$.

(Note that the smallest joint refinement of $[\Gamma_1 : \Delta_1]$ and $[\Gamma_2 : \Delta_2]$, if there is any such refinement, is $[\Gamma_1 \cup \Gamma_2 : \Delta_1 \cup \Delta_2]$.)

FACT 3: *A set $\mathcal{D}$ finite positions is directed if and only if there is some position $[\Gamma : \Delta]$ (finite or infinite) such that $\mathcal{D}$ is the set of finite positions refined by $[\Gamma : \Delta]$.*

This position $[\Gamma : \Delta]$ is said to be the LIMIT of the directed set $\mathcal{D}$.

Proof: Given a position P, if $\mathcal{D}$ the set of finite positions refined by P, this set is clearly closed downward under refinement, and it contains upper bounds, so it is

[14] Readers familiar with tableaux proofs will recognise the construction [1, 13].

directed. Conversely, if $\mathcal{D}$ is some directed class, define the limit position $[\Gamma : \Delta]$ of $\mathcal{D}$ in the obvious way: Γ is the set of all formulas or terms occuring in the LHS of some position in $\mathcal{D}$, and Δ is the set of all formulas or terms occuring in the RHS of some position in $\mathcal{D}$. Let $P = [L_1, \ldots, L_n : R_1, \ldots, R_m]$ be some finite position refined by $[\Gamma : \Delta]$. We want to show that P is in $\mathcal{D}$. Each L_i is a formula or term occuring in Γ and each R_j is a formula or term occuring in Δ. This means that each L_i occurs in the LHS of some position in $\mathcal{D}$ and each R_j occurs in the RHS of some position in $\mathcal{D}$. Since $\mathcal{D}$ is directed, it is closed downward under refinement, so the atomic positions $[L_i :]$ and $[: R_j]$ are in $\mathcal{D}$, so their finite upper bound, namely P, is in $\mathcal{D}$ too. ∎

FACT 4: *For any sequence of positions* $[\Gamma_1 : \Delta_1]$, $[\Gamma_2 : \Delta_2], \ldots$ *where each* $[\Gamma_{i+1} : \Delta_{i+1}]$ *is a refinement of* $[\Gamma_i : \Delta_i]$, *the set* $\mathcal{D}$ *of all finite positions refined by some* $[\Gamma_i : \Delta_i]$ *in the sequence is the directed set with limit* $[\bigcup_i^\infty \Gamma_i : \bigcup_i^\infty \Delta_i]$.

Proof: If $\Gamma' \subseteq \bigcup_i^\infty \Gamma_i$ and $\Delta' \subseteq \bigcup_i^\infty \Delta_i$ are finite, there must be some n where $\Gamma' \subseteq \Gamma_n$ and $\Delta' \subseteq \Delta_n$. Since $[\Gamma_n : \Delta_n]$ is a position, it follows that $[\Gamma' : \Delta']$ is a position too, and since Γ' and Δ' were arbitrary finite subsets of $\bigcup_i^\infty \Gamma_i$ and $\bigcup_i^\infty \Delta_i$ respectively, it follows that $[\bigcup_i^\infty \Gamma_i : \bigcup_i^\infty \Delta_i]$ is indeed a position. So, the set of all finite positions refined by $[\bigcup_i^\infty \Gamma_i : \bigcup_i^\infty \Delta_i]$ is a directed set, and these are exactly the positions refined by some position in the sequence. ∎

In such a construction, we call the directed set $\mathcal{D}$ the WAKE of the sequence $[\Gamma_i : \Delta_i]$ of finite positions.

We will end this section with he characterisation of a construction of a special sequence of positions, with the target, in the limit, a *fully refined position*.

A sequence of finite positions $[\Gamma_1 : \Delta_2]$, $[\Gamma_2 : \Delta_2], \ldots$ is said to be FULLY REFINING if for each member $[\Gamma_i : \Delta_i]$ in the sequence, and each formula or term occuring in Γ_i or Δ_i, the an appropriate *refinement* of the position for that formula or term (a refinement occuring in Figure 1, 2 or 3) is refined by some other member $[\Gamma_j : \Delta_j]$ of the sequence.

FACT 5: *The following algorithm constructs a fully refining sequence of positions starting from a finite position* $[\Gamma : \Delta]$*:*

1. Enumerate each sentence or term in the starting position $[\Gamma : \Delta]$. The number for each sentence or term will be the order in which it is processed for refinement. Keep track of the next available number.
2. Form the finite set T of terms that occur somewhere in $[\Gamma : \Delta]$, either directly as members of Γ or Δ, or as components of formulas or terms occuring in formulas in Γ and Δ.
3. Reserve an infinite stock N of $\{n_1, n_2, \ldots\}$ from the language $\mathcal{L}$ which do not appear in T.

4. Now we start the loop. Take the *first* available formula in our enumeration.

   (a) If it is a negation $\neg A$ in the LHS, form the next position by adding A to the RHS, assign A the next available number, mark $\neg A$ as unavailable and repeat.

   (b) If it is a negation $\neg A$ in the RHS, form the next position by adding A to the LHS, assign A the next available number, mark $\neg A$ as unavailable and repeat.

   (c) If it is a conjunction $A \wedge B$ in the LHS, form the next position by adding A and B to the LHS, assign A and B the next available numbers, mark $A \wedge B$ as unavailable and repeat.

   (d) If it is a conjunction $A \wedge B$ in the RHS, then either a position is formed by adding A or by adding B to the RHS. Choose whichever works, and that formula to the RHS, assign it the next available number, mark $A \wedge B$ as unavailable and repeat.

   (e) If it is a disjunction $A \vee B$ in the LHS, then either a position is formed by adding A or by adding B to the LHS. Choose whichever works, and that formula to the LHS, assign it the next available number, mark $A \vee B$ as unavailable and repeat.

   (f) If it is a disjunction $A \vee B$ in the RHS, form the next position by adding A and B to the RHS, assign A and B the next available numbers, mark $A \vee B$ as unavailable and repeat.

   (g) If it is a conditional $A \supset B$ in the LHS, then either a position is formed by adding A to the RHS or by adding B to the LHS. Choose whichever works, and that formula, assign it the next available number, mark $A \supset B$ as unavailable and repeat.

   (h) If it is a disjunction $A \supset B$ in the RHS, form the next position by adding A to the LHSand B to the RHS, assign A and B the next available numbers, mark $A \supset B$ as unavailable and repeat.

   (i) If is a definedness statement $t\downarrow$ on either side, form the next position by adding the term t to the same side, assign this term the next available number, mark $t\downarrow$ as unavailable, and repeat.

   (j) If it is a universal quantifier $(\forall x)A(x)$ in the LHS, then do this for each term t in T—either adding $A(t)$ to the, or t to the RHS results in a position, so add whichever works, tagging the added formula or term with the next available number. Mark $(\forall x)A(x)$ as unavailable, but add it to the *reserve* list, to be processed whenever new terms are added to T.

(k) If it is a universal quantifier $(\forall x)A(x)$ in the RHS, then remove the first name from the list N of fresh names, add it to the set T of used terms, and form a new position by adding n LHS of the position and adding $A(n)$ to the RHS, tagged with the next available number, and mark $(\exists x)A(x)$ as complete. Then revisit each formula on the *reserve* list as follows:

i. For each universal quantifier $(\forall x)B(x)$ in the LHS, add $B(n)$ to the LHS, assigned the next available number.

ii. For each existential quantifier $(\exists x)C(x)$ in the RHS, add $C(n)$ to the RHS, assigned the next available number.

(l) If it is an existential quantifier $(\exists x)A(x)$ in the LHS, then remove the first name from the list N of fresh names, add it to the set T of used terms, and form a new position by adding n and $A(n)$ to the LHS, tagging $A(n)$ with the next available number, and marking $(\exists x)A(x)$ as unavialable. Then revisit each formula on the *reserve* list as follows:

i. For each universal quantifier $(\forall x)B(x)$ in the LHS, add $B(n)$ to the LHS, assigned the next available number.

ii. For each existential quantifier $(\exists x)C(x)$ in the RHS, add $C(n)$ to the RHS, assigned the next available number.

(m) If it is an existential quantifier $(\exists x)A(x)$ in the RHS, then do this for each term t in T—either adding $A(t)$ to the RHS, or t to the RHS results in a position, so add whichever works, tagging the added formula or term with the next available number. Mark $(\exists x)A(x)$ as unavailable, but add it to the *reserve* list, to be processed whenever new terms are added to T.

(n) If it is a primitive predication $Ft_1 \cdots t_n$ in the LHS, add $t_1, \ldots, t_n$ to the LHS, tagging each term with the next available numbers, and marking $Ft_1 \cdots t_n$ as unavailable.

(o) If it is a function term $f(t_1, \ldots, t_n)$ in the LHS, add $t_1, \ldots, t_n$ to the LHS, tagging each term with the next available numbers, and marking $f(t_1, \ldots, t_n)$ as unavailable.

(p) If it is a primitive predication or a function term in the RHS, mark it as unavailable immediately, and repeat.

5. End only when there is no formula available in the enumeration.

Proof: At each stage of the process, the result is another position, refining the previous positions. At each stage of the process, only finitely many formulas or terms are added, to construct the new position, so each formula or term added to a position along the way is assigned a finite number, except for fresh names added to the stock of defined terms—they get no numbers because they do not need to

be processed. The process then considers each formula or term in the order of the number assigned, and ensures that we add the relevant formulas or terms in order to construct a position closed under the appropriate refinement condition. The result is a fully refining sequence of positions. The limit of such a sequence is a *fully refinied position* which refines the starting position.

6 FULLY REFINED POSITIONS AND MODELS

A fully refined position $[\Gamma : \Delta]$ is very special indeed. We will use such positions to motivate the definition of *models* for our logic.

A MODEL for the logic DL is a structure $\mathfrak{M}$ consisting of

1. A *domain* D.
2. An n-ary predicate F is interpreted as a subset $F^{\mathfrak{M}}$ of D^n (as usual).
3. An n-ary function symbol f is interpreted as a *partial function* $f^{\mathfrak{M}} : D^n \rightharpoonup D$.

Given an assignment α of values to variables, we recursively define the interpretation partial function $[\![\cdot]\!]_{\mathfrak{M},\alpha}$ assigning values to terms as follows:[15]

- $[\![x]\!]_{\mathfrak{M},\alpha} = \alpha(x)$
- $[\![f(t_1,\ldots,t_n)]\!]_{\mathfrak{M},\alpha} = f^{\mathfrak{M}}([\![t_1]\!]_{\mathfrak{M},\alpha},\ldots,[\![t_n]\!]_{\mathfrak{M},\alpha})$ if each $[\![t_i]\!]_{\mathfrak{M},\alpha}$ is defined, and $f^{\mathfrak{M}}$ is defined on the inputs $[\![t_1]\!]_{\mathfrak{M},\alpha},\ldots,[\![t_n]\!]_{\mathfrak{M},\alpha}$.

Then given the interpretation of terms, we define the interpretation of *sentences* or *open formulas* (sentences with some quantifiers removed, exposing variables) relative to an assignment of values to variables, as follows:

- $\mathfrak{M} \vDash_\alpha t{\downarrow}$ iff $[\![t]\!]_{\mathfrak{M},\alpha}$ is defined.
- $\mathfrak{M} \vDash_\alpha Ft_1\cdots t_n$ iff for each i, the value $[\![t_i]\!]_{\mathfrak{M},\alpha}$ is defined, and the n-tuple $\langle [\![t_n]\!]_{\mathfrak{M},\alpha},\ldots,[\![t_n]\!]_{\mathfrak{M},\alpha}\rangle \in F^{\mathfrak{M}}$
- $\mathfrak{M} \vDash_\alpha A \wedge B$ iff $\mathfrak{M} \vDash_\alpha A$ and $\mathfrak{M} \vDash_\alpha B$.
- $\mathfrak{M} \vDash_\alpha A \vee B$ iff $\mathfrak{M} \vDash_\alpha A$ or $\mathfrak{M} \vDash_\alpha B$.
- $\mathfrak{M} \vDash_\alpha A \supset B$ iff $\mathfrak{M} \nvDash_\alpha A$ or $\mathfrak{M} \vDash_\alpha B$.
- $\mathfrak{M} \vDash_\alpha \neg A$ iff $\mathfrak{M} \nvDash_\alpha A$.
- $\mathfrak{M} \vDash_\alpha (\forall x)A(x)$ iff $\mathfrak{M} \vDash_{\alpha[x:=d]} A(x)$ for every d in D.

[15] In the special case where D is *empty*, which we allow, there are *no* assignments of values to variables, all predications are false, all function symbols and terms are undefined, and the interpretation function is trivial, with all existentially quantified formulas false and all universally quantified formulas true.

- $\mathfrak{M} \vDash_\alpha (\exists x)A(x)$ iff $\mathfrak{M} \vDash_{\alpha[x:=d]} A(x)$ for some d in D.

We say that a model $\mathfrak{M}$ is a MODEL OF THE POSITION $[\Gamma : \Delta]$ if and only if every sentence in Γ is true in $\mathfrak{M}$, every term in Γ is defined in $\mathfrak{M}$, every sentence in Δ is false in $\mathfrak{M}$ and every term in Δ is undefined in $\mathfrak{M}$.

FACT 6: *For any fully refined position* $[\Gamma : \Delta]$*, a model where — (1) the domain* D *consists of the terms occuring as members of* Γ*, such that (2) every* n*-ary predicate* F *is interpreted as the set of* n*-tuples* $\langle t_1, \ldots, t_n\rangle$ *where* $Ft_1 \cdots t_n$ *is in* Γ*, and (3) where the* n*-ary function symbol* f *is interpreted by setting* $f(t_1, \ldots, t_n)$ *to be defined if and only if the term occurs in* Γ*, and then in that case takes* itself *as its value — is indeed a model of the position* $[\Gamma : \Delta]$.

Proof: We prove this first for terms and then sentences in the position. For terms, the result is nearly immediate. Any term in Γ is defined and has itself as its value. The closure of $[\Gamma : \Delta]$ under function application ensures that $f(t_1, \ldots, t_n)$ is defined (in the LHS) only when its components $t_1, \ldots, t_n$ is defined (in the LHS).

For sentences, if $Ft_1 \cdots t_n$ is in Γ, then since the position is predicate refined, each t_i is defined in the model and has itself as its value. The interpretation of $Ft_1 \cdots t_n$ in the model assigns it to be true. If it is in Δ, then it is in Γ (lest $[\Gamma : \Delta]$ fail to be a position), so it is not true in the model.

If $t\downarrow$ is in Γ, then by refining under definedness, t is in Γ, so t is defined in the model and hence $t\downarrow$ is true in the model. Similarly, if $t\downarrow$ is in Δ, then t is in Δ, so t cannot be defined in the model (it is defined only when t is in Γ, but $[\Gamma : \Delta]$ is a position, so we cannot have t in Γ), so $t\downarrow$ is false in the model.

For $A \wedge B$ in Γ, by refinedness under conjunction, A and B are in Γ, and by hypothesis, they are true in the model. So, $A \wedge B$ is true in the model. If $A \wedge B$ is in Δ, by refinedness under conjunction, either A or B is in Δ, so by hypothesis,one is false in the model. So, $A \wedge B$ is false in the model. The cases for the other propositional connectives are similar.

For $(\forall x)A(x)$ in Γ, for every term t occuring in the position, either t is in Δ, and hence is not defined in the model, or $A(t)$ is in Γ and hence, is true in the model. It follows that for every object t in the domain, $A(t)$ is true in the model. It follows that $A(x)$ is true in the model whenever x is assigned the value t, and hence, that $(\forall x)A(x)$ is true in the model, too. If $(\forall x)A(x)$ is in Δ, then for some name n we have n in Γ (and hence, n is defined in the model) and $A(n)$ in Δ. By hypothesis, $A(n)$ is false in the model, and hence, $A(x)$ is false in the model when x is assigned the value n, so $(\forall x)A(x)$ is false in the model.

The case for the existential quantifier is similar, and that ends the proof. This model constructed from the fully refined position is a model of that position. ∎

7 COMPLETENESS AND CUTS

The fact we have just proved gives us a COMPLETENESS for the sequent calculus with respect to the models. We have shown that for any underivable sequent, there

is a model (a model constructed out of a fully refined extension of that sequent) in which that sequent fails in the sense that it takes the LHS of the sequent to be true (defined) and the RHS to be false (undefined).

However, we have proved much more than the completeness of the sequent calculus. We have also shown that any derivation in DL[*LR*, *Cut*] can be derived in DL[*LR*] without making use of either sentence or term *Cuts*. Here is why. We have already shown

FACT 7: *Any finite position in* DL[*LR*] *is refined by some* fully refined *position.*

Fact 5 shows us how to extend any finite DL[*LR*] position by a fully refining sequence, whose limit is a fully refined position.

FACT 8: *Any finite* DL[*LR*] *position has some model.*

Fact 6 shows us that a model defined from a fully refined DL[*LR*] position is a model of that position. Since by Fact 7, any finite DL[*LR*] position is refined by a fully refined position, and since a model of a position is also a model of any position that *refines to* that position, any finite DL[*LR*] position has some model.

On the other hand, it is straightforward to prove a *soundness* theorem. We can show that *derivable* sequents $\Gamma \succ \Delta$ *hold* in all models in the sense that in any model $\mathfrak{M}$ if each element of Γ is true (for sentences) or defined (for terms) then some element of Δ is true (for sentences) or defined (for terms). In fact, this holds for sequents derivable not only in DL[*LR*], but also DL[*LR*, *Cut*].

FACT 9: *If* $\Gamma \succ \Delta$ *is derivable in* DL[*LR*, *Cut*] *then it holds in all models* $\mathfrak{M}$.

Proof: A straightforward induction on the structure of the derivation. IDENTITY axioms hold trivially: if A is true, it is true. If t is defined, it is defined. The structural rules—including *Cuts*—are straightforward. If $\Gamma \succ t, \Delta$ and $\Gamma, t \succ \Delta$ both hold in $\mathfrak{M}$ then by $\Gamma \succ t, \Delta$ if each element of Γ is true then either an element of Δ is true (in which case $\Gamma \succ \Delta$ holds in $\mathfrak{M}$) or t is defined in $\mathfrak{M}$. In that case, since $\Gamma, t \succ \Delta$ holds in $\mathfrak{M}$, in any case an element of Δ is true, and so, $\Gamma \succ \Delta$ holds in $\mathfrak{M}$ regardless. The case for sentence *Cut* has the same structure.

The connective, predication, function application, definedness and quantifier rules are satisfied trivially by way of the truth conditions for formulas, and the result is straightforward to prove. ∎

It follows then, that we have the admissibility of *Cut*.

FACT 10: *If* $\Gamma \succ \Delta$ *is derivable in* DL[*LR*, *Cut*], *then it is derivable in* DL[*LR*] *too.*

Proof: If $\Gamma \succ \Delta$ were not derivable in DL[*LR*] then $[\Gamma : \Delta]$ would be a DL[*LR*] position. Then Fact 8 tells us that this position has some model, a model in which each element of Γ is true (defined) and each element of Δ is false (undefined). This means that $\Gamma \succ \Delta$ cannot be derived in DL[*LR*, *Cut*] either. ∎

8 CONSEQUENCES AND QUESTIONS

I will end with some brief observations and questions for further exploration.

First and foremost, this paper has explored *defining rules* as a new way to answer Prior's question concerning the difference between rules for logical constants familiar to us, like conjunction, and defective rules like Prior's rules for *tonk*. We have seen that this treatment of defining rules naturally extends to quantifiers, once we have not only the syntax of the predicate/term distinction, but also the notion of a *deductively general* term. This notion seems to be deeply embedded in proof-theoretical treatments of the quantifiers, and it seems well suited to the normative pragmatic intepretation of the sequent calculus in terms of norms governing assertion and denial.

If we allow for disagreement not only on what we take to be *true* or *false*, but also on what there *is*, then it it seems very natural to extend the normative pragmatic treatment of assertion and denial to *pro* and *con* attitudes to *terms*, as well. The result is a relatively straightforward sequent calculus for a logic in which terms are free of existential import. This sequent calculus has pleasing theoretical properties, and natural *Left* and *Right* rules for the quantifiers can be understood as arising out of simple *defining* rules in a straightforward way. The fact that the results are so straightforward—using techniques known from elsewhere [10, 11], with small changes to incorporate the behaviour of deductively general terms and the rule of *Specification*—lends some support to the thought that defining rules play a useful part in the design of a proof-theoretical framework.

However, questions remain. In following papers I will address questions concerning *modality* and *identity*. Both are very important if the issue of the semantics of quantifiers and the behaviour of non-denoting terms is to be thoroughly understood. We have focussed here on one motivation for non-denoting terms: from mathematics. The behaviour of *contingently* non-denoting terms seems to be very different, but these techniques seem like they may be appropriate there, too. But are they? The proof of this will be in the detail, and there is more detail than could reasonably fit in this paper, hence a sequel.

Before that sequel, however, I will end with a question, which raises concerns for the significance of the choice of the free logic for quantifiers, and of the kind of criterion for meaningfulness that may be provided by defining rules in a logical system. Suppose we take DL, definedness logic seriously. Consider the following defining rules for things that certainly *look* like quantifiers:

$$\frac{\Gamma \succ A(n), \Delta}{\Gamma \succ (\Pi x)A(x), \Delta}\,[\Pi Df] \qquad \frac{\Gamma, A(n) \succ \Delta}{\Gamma, (\Sigma x)A(x) \succ \Delta}\,[\Sigma Df]$$

These have the same shape as quantifier rules, except the *definedness* conditions have been left out. These *are* defining rules. Do they define concepts? Do they define meaningful concepts? Logical concepts? If so, in what sense is it appropriate

to infer $(\Sigma x)\neg x\downarrow$ from $\neg\frac{1}{0}\downarrow$? Could this provide the beginnings of a new kind of defence of Meinong's quantifiers [4, 6, 7, 12]?

REFERENCES

[1] IRVING H. ANELLIS. "From semantic tableaux to Smullyan trees: a history of the development of the falsifiability tree method". *Modern Logic*, 1(1):36–69, 1990.

[2] ARNON AVRON. "Gentzen-type systems, resolution and tableaux". *Journal of Automated Reasoning*, 10(2):265–281, 1993.

[3] NUEL D. BELNAP. "Tonk, Plonk and Plink". *Analysis*, 22:130–134, 1962.

[4] FRANCESCO BERTO. ""There Is an 'Is' in 'There Is'": Meinongian Quantification and Existence". In A. TORZA, editor, *Quantifiers, Quantifiers, and Quantifiers*, Synthese Library. Springer, 2015.

[5] SOLOMON FEFERMAN. "Definedness". *Erkenntnis*, 43(3):295–320, 11 1995.

[6] ALEXIUS MEINONG. *On Assumptions*. University of California Press, Berkeley and Los Angeles, California, 1983. Translated and edited by James Heanue.

[7] GRAHAM PRIEST. *Doubt Truth to be a Liar*. Oxford University Press, 2006.

[8] ARTHUR N. PRIOR. "The Runabout Inference-Ticket". *Analysis*, 21(2):38–39, 1960.

[9] GREG RESTALL. "Multiple Conclusions". In PETR HÁJEK, LUIS VALDÉS-VILLANUEVA, AND DAG WESTERSTÅHL, editors, *Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*, pages 189–205. KCL Publications, 2005. http://consequently.org/writing/multipleconclusions.

[10] GREG RESTALL. "Truth Values and Proof Theory". *Studia Logica*, 92(2):241–264, 2009. http://consequently.org/writing/tvpt/.

[11] GREG RESTALL. "A Cut-Free Sequent System for Two-Dimensional Modal Logic, and why it matters". *Annals of Pure and Applied Logic*, 163(11):1611–1623, 2012. http://consequently.org/writing/cfss2dml/.

[12] RICHARD ROUTLEY. *Exploring Meinong's Jungle*. ANU, 1979.

[13] R. M. SMULLYAN. *First-Order Logic*. Springer-Verlag, Berlin, 1968. Reprinted by Dover Press, 1995.

[14] PHILIP WADLER. "Call-by-value is dual to call-by-name, reloaded". In *Rewriting Technics and Application, RTA'05*, volume 3467 of *Lecture Notes in Computer Science*, pages 185—203. Springer, 2005.