

A GRAPH FORMALISM FOR PROOFS IN THE LAMBEK CALCULUS WITH
PRODUCT

by

Timothy Alexander Dalton Fowler

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Copyright © 2006 by Timothy Alexander Dalton Fowler

Abstract

A graph formalism for proofs in the Lambek Calculus with Product

Timothy Alexander Dalton Fowler

Master of Science

Graduate Department of Computer Science

University of Toronto

2006

Since the introduction of the Lambek calculus in Lambek (1958), there has been a great deal of interest in its usefulness as a grammar for parsing in natural language. Several variants have been introduced and for each, questions of tractability and usefulness have been posited and answered. Pentus (2003) answered the question of tractability of the original calculus by providing an NP-completeness proof.

The simplest of these variants is the version of the original calculus without the product, the computational complexity of which is as yet unknown. This thesis seeks to identify the precise implications of products on the complexity of parsing in the calculus. Towards this goal, a graph formalism for proofs in the original calculus is extended from the work in Penn (2001).

We then present a simplified, graphical NP-completeness proof for derivability in the Lambek calculus with product and consider the potential intractability of the Lambek calculus without product.

Dedication

For those who knew I could do this.

Acknowledgements

First, I must acknowledge my thesis advisor Gerald Penn. All of the work in this thesis either builds on his previous work or was heavily guided by him. Without his knowledge of mathematics, linguistics and computer science, this work would simply not have been possible. His attentiveness and encouragement were absolutely necessary to have made the progress that I have.

Second, I have to thank all of those who have been near me in my time in Toronto. This includes the people at Massey college, the floppy discs, the floating points, the computational linguistics group, the Monday night poker group, the guys of torontula, my roommates, my friends and my companions. Also, my little friends deserve thanks because of the peace they have brought me including Jesus, Alex, Spidey, Abe, the vine and Mr. Gecko.

Thirdly, I should thank my second reader Michael Molloy whose comments provided much needed guidance.

Finally, I want to thank my family for giving me the environment in which to begin, for continuing to support me well into adulthood and for never questioning my decisions.

Contents

1	Introduction	1
1.1	Statement of thesis and objectives	2
1.1.1	Thesis	2
1.1.2	Objectives	3
1.2	Structure of the document	4
2	The Lambek Calculus	6
2.1	The Lambek Calculus	6
2.1.1	A Formal Definition	7
2.1.2	The Lambek Calculus and <i>CUT</i>	10
2.2	Complexity results for the Lambek Calculus	10
3	Proof nets for the Lambek Calculus	12
3.1	Proof nets	12
3.1.1	Proof structures	13
3.1.2	Girard style correctness	17
3.1.3	Roorda style correctness	21
4	LC Graphs for L^\bullet	30
4.1	Some additional terminology for proof nets	30
4.2	LC Graphs	31

4.3	Basic Properties of LC Graphs	32
4.4	Unitary	34
4.4.1	Definitions	34
4.4.2	Basic Theorems	38
4.4.3	Reproving 5.3.4 and 5.3.6 of Roorda (1991)	40
4.5	Derivability conditions	45
4.6	I is complete and sound	46
4.6.1	Soundness	46
4.6.2	Completeness	47
4.6.3	Equivalence of PN(CT) and I(CT)	52
5	LC Graphs and the NP Completeness of L^\bullet	54
5.1	The reduction	54
5.2	Correctness of the reduction	56
5.2.1	Proof nets of $E_1(t_1), \dots, E_n(t_n) \vdash G$	57
5.2.2	The LC graph for $E_1(t_1), \dots, E_n(t_n) \vdash G$	60
6	Conclusion	68
6.1	Future directions	69
	Bibliography	70

List of Tables

2.1	Four variants of the Lambek Calculus.	7
3.1	Three links of proof structures for Girard style correctness.	17
3.2	Translation from proof structures to DR -graphs.	18
3.3	Translation from proof structures to $R\&B$ -graphs.	20
3.4	Substitutions due to positive polarity rules.	25
5.1	The two coats of the reduction.	55

List of Figures

2.1	Sequents.	8
2.2	Axioms in the Lambek Calculus.	8
2.3	Inference rules in the Lambek Calculus.	9
2.4	Proof that the sequent $(A/A) \vdash ((A/A) \bullet A)$ is in L^\bullet	9
2.5	Proof that the sequent $((A/(A \setminus A))/A) \vdash A \setminus (A \setminus A)$ is in L^*	9
2.6	The <i>CUT</i> rule.	10
3.1	Terminal formulae for the sequent $\alpha_1, \dots, \alpha_n \vdash \alpha$	13
3.2	Proof frame building rules.	14
3.3	The proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$	15
3.4	The proof frame for $(A/A) \vdash ((A/A) \bullet A)$	15
3.5	A proof structure for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$	15
3.6	A proof structure for $(A/A) \vdash ((A/A) \bullet A)$	16
3.7	Another proof structure for $(A/A) \vdash ((A/A) \bullet A)$	16
3.8	A proof structure with a non-planar lineage for $(A/A) \vdash ((A/A) \bullet A)$	16
3.9	The <i>DR</i> -graph for the proof structure in figure 3.5.	19
3.10	The <i>DR</i> -graph for the proof structure in figure 3.6.	19
3.11	The <i>DR</i> -graph for the proof structure in figure 3.7.	19
3.12	The <i>R&B</i> -graph for the proof structure in figure 3.5.	21
3.13	The <i>R&B</i> -graph for the proof structure in figure 3.6.	21
3.14	The <i>R&B</i> -graph for the proof structure in figure 3.7.	22

3.15	Rules for annotating proof frames with lambda terms.	23
3.16	Lambda terms annotated to the proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$	24
3.17	Lambda terms annotated to the proof frame for $(A/A) \vdash A \vdash ((A/A) \bullet A)$	24
3.18	Variable substitution of a proof structure for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$	25
3.19	Variable substitution of a proof structure for $(A/A) \vdash A \vdash ((A/A) \bullet A)$	26
3.20	Variable substitution of another proof structure for $(A/A) \vdash A \vdash ((A/A) \bullet A)$	26
3.21	A proof structure for $((A/(A \setminus A))/A) \vdash A \vdash (A \setminus A) \vdash (A \setminus A) \vdash A$	28
3.22	The LC graph for the proof structure in figure 3.21.	28
3.23	Another proof structure for $((A/(A \setminus A))/A) \vdash A \vdash (A \setminus A) \vdash (A \setminus A) \vdash A$	28
3.24	The LC graph for the proof structure in figure 3.23.	29
4.1	The proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$	31
4.2	The LC Graph for the proof structure in 3.18.	32
4.3	The proof frame for $(A/A) \vdash (A/(A \bullet A)) \bullet A$ (repeat of figure 4.1).	49
5.1	The proof frame for $E_i^0(t) = p_{i-1}^0 \setminus p_i^0$	57
5.2	The proof frame for $(p_{i-1}^j \setminus E_i^{j-1}(t)) \bullet p_i^{j-}$	57
5.3	The proof frame for $p_{i-1}^j \setminus (E_i^{j-1}(t) \bullet p_i^j)^-$	57
5.4	The proof frame for $G^0 = p_0^0 \setminus p_n^0$	58
5.5	The proof frame for $G^j = (p_0^j \setminus G^{j-1}) \bullet p_n^{j-}$	58
5.6	LC graph of substitutions of $E_1(t_1), \dots, E_n(t_n) \vdash G$	60
5.7	Out-neighbourhoods of b^j for $0 \leq j \leq m$	61
5.8	Out-neighbourhoods of a_i^j for $0 \leq j \leq m$	61
5.9	LC graph of $E_1(t_1), \dots, E_n(t_n) \vdash G$	63
5.10	Simplified view of the LC graph of $E_1(t_1), \dots, E_n(t_n) \vdash G$	65
5.11	Proof structure of $E_1(0), E_2(0) \vdash G$ for $x_1 \vee \neg x_2$	66
5.12	Proof structure of $E_1(0), E_2(1) \vdash G$ for $x_1 \vee \neg x_2$	66
5.13	LC graph of $E_1(0), E_2(0) \vdash G$ for $x_1 \vee \neg x_2$ (see in figure 5.11).	67

5.14 LC graph of $E_1(0), E_2(1) \vdash G$ for $x_1 \vee \neg x_2$ (shown in figure 5.12). 67

Chapter 1

Introduction

Lambek (1958) introduced the Lambek calculus as both a categorial grammar and a sub-structural logic. The Lambek calculus treats the natural language recognition problem as equivalent to a logical sequent derivability problem. This calculus has been well studied and systems closely related to it are widely used in natural language processing such as combinatory categorial grammar. Its primary motivation comes from the ease of meaning extraction from its parse trees which is not available from context-free grammars.

Since its introduction, many variants of the Lambek calculus have been introduced including commutative variants, non-associative variants and variants with more or less logical connectives. Work has been done on the computational complexity of the calculus and its variants with varying degrees of success.

Pentus (2003) proved that sequent derivability in the original calculus is NP complete. However, by removing a single logical connective, the product, we arrive at a simpler system. The sequent derivability question for this logic is as yet unknown. In addition, the applications to natural language are not greatly reduced by this simplification due to the lack of applications of the product in practice. For some linguistic motivations of product see Wood (1988), Morrill and Gavarro (1992) and Morrill (1990). They are generally limited to the syntax-morphology interface.

Girard (1987) introduced linear logic and it was quickly realized that the Lambek calculus is equivalent to a fragment of linear logic known as the non-commutative multiplicative fragment. In addition to his linear logic, Girard introduced a method for asserting the existence of proofs in the multiplicative fragment without actually searching through proofs, which he called proof nets. Since their introduction, much work has been done on the presentation of proof nets, and various graph representations of proof nets have been proposed and analyzed (see Danos and Regnier (1989), Roorda (1991) and Penn (2001)).

Proof nets provide a drastic reduction in the search space for the sequent derivability problem by ignoring spurious ambiguities. It is this fact that makes proof nets especially relevant to our quest for tractability. It should be noted that we are specifically looking at sequent derivability in the calculus, which is a decision problem. In contrast, the search for, or enumeration of, all possible proofs of a sequent is necessarily exponential.

Penn (2001) introduced a particular graph formalism for proof nets called LC graphs which only specifies such graphs for the Lambek calculus without product. Our goal is then to extend these graphs to the Lambek calculus with product to determine as precisely as possible the difference between the two calculi in terms of parsing complexity.

1.1 Statement of thesis and objectives

1.1.1 Thesis

There is a fundamental difference between the Lambek calculus with product and the Lambek calculus without product that may determine tractability. This difference can be seen in the form of a single path condition on graph representations of proofs in the calculi.

1.1.2 Objectives

The objectives of this document are fourfold:

1. Graph representations of proof nets:

This document sets out to compare and contrast the various graph representations of proof nets such as those in Girard (1987), Danos and Regnier (1989), Roorda (1991), Lamarche (1995), Retore (1996), Penn (2001) and Carpenter and Morrill (2005).

They are categorized according to their similarities so that their strengths and weaknesses can easily be seen.

2. The difference between the Lambek calculus without product (L) and the Lambek calculus with product (L^\bullet):

This document seeks to identify the precise difference in the sequent derivability problems of L and L^\bullet . To this end, LC graphs for L^\bullet are introduced in a very similar way to the LC graphs for L introduced in Penn (2001). The result is elegant in that the only difference is a single path condition on these graphs.

3. NP completeness proof for L^\bullet :

This document partially reproves the NP completeness result of Pentus (2003) using an approach based heavily on LC graphs for L^\bullet . The question is posed as to whether this approach can be modified to provide a proof that sequent derivability in L is NP complete and is answered in the negative.

4. Literature Review:

The final purpose of this thesis is to integrate the numerous and myopically presented results in this field. The poor level of scholarship in this area has led to a very high level of redundancy and confusion and doubtlessly shares much of the

responsibility for our lack of progress in understanding the Lambek Calculus in the roughly fifty years since its introduction. Much of the work in this area has borne a strong resemblance to previous work without references of any kind, leading to numerous solutions and terminological systems for the same problem without any consolidation.

1.2 Structure of the document

Chapter 2: The Lambek Calculus

This chapter introduces the Lambek calculus as originally stated in Lambek (1958). In addition, the known complexity results for these calculi are presented.

Chapter 3: Proof nets for the Lambek calculus

This chapter introduces proof nets as originally presented by Girard (1987). A number of graph-theoretic representations of these nets have been proposed and several of them are presented here.

Chapter 4: LC Graphs for L^\bullet

A graph-theoretic representation of proof nets is introduced based on the work in Penn (2001). The primary motivation here is to isolate the important difference between L and L^\bullet by studying their LC graphs. The latter half of the chapter is a correctness proof of the representation.

Chapter 5: LC Graphs and the NP-completeness of L^\bullet

This chapter consists of a reproof of the NP-completeness result of Pentus (2003). It is expected that the proof given here is easier to understand because it is largely presented from a graphical standpoint. In addition, the importance of the well-formedness condition called $I(4)$ is brought to bear.

Chapter 6: Conclusion

This chapter gives some conclusions and some future research possibilities.

Chapter 2

The Lambek Calculus

The Lambek Calculus is both a logic and a grammar formalism originally introduced by Lambek (1958). Among other logics, the Lambek Calculus is relatively weak and is classified as a sub-structural logic. It is a logic in the sense that it is characterized by logical derivability from a set of axioms and a set of inference rules. It is sub-structural in the sense that it does not have the inference rules for weakening, contraction or exchange found in classical first-order logic. The original Lambek Calculus is both non-commutative and associative, although commutative and non-associative variants appear in the literature.

The primary motivation of the Lambek Calculus is as a grammar formalism for parsing in natural language. Extensive work has been done on the linguistic applications of categorial grammar which is a class of grammar formalisms which includes the Lambek Calculus.. A good representative of this work can be seen in Steedman (2000) and Carpenter (1997).

2.1 The Lambek Calculus

Along with its original formulation, the Lambek calculus has seen several variants proposed. The original form of the calculus in Lambek (1958) included three logical con-

nectives: left implication ($/$), right implication (\backslash) and product (\bullet). Also, sequents were prohibited from having empty antecedents or, more technically, empty premise derivations were prohibited. More recently, two other variants have been shown to be of interest because of their relation to the original logic (see Roorda (1991), Pentus (1997), Penn (2001)) as shown in table 2.1.

Table 2.1: Four variants of the Lambek Calculus.

	Allows Products	Product Free
Prohibits Empty Premises	$L^{\bullet*}$	L^*
Allows Empty Premises	L^{\bullet}	L

These particular calculi are interesting because they are likely to sit on the threshold of tractable sequent derivability as will be discussed in section 2.2. Several other variants have been defined in the literature and some will be discussed briefly in this thesis.

2.1.1 A Formal Definition

The Lambek Calculus, either interpreted as a logic or as a grammar, is a set of sequents which is a proper subset of all possible sequents. To begin with, we need a set of Lambek atoms A, B, C, \dots . Lambek atoms correspond to the notion of basic categories in categorial grammar.

Definition 2.1.1. A *formula* in the Lambek Calculus is defined as follows:

- If A is a Lambek atom then A is a formula.
- If α and β are formulae, then (α/β) , $(\alpha\backslash\beta)$ and $(\alpha\bullet\beta)$ are formulae.

Definition 2.1.2. The *largest scoping connective* of (α/β) (respectively $(\alpha\backslash\beta)$ and $(\alpha\bullet\beta)$) is $/$ (respectively \backslash and \bullet).

Definition 2.1.3. A *sequent* is defined as $\Gamma \vdash \alpha$ where Γ is a sequence of formulae and α is a formula.

Figure 2.1 gives two possible sequents. Throughout this thesis, capital Latin letters (A, B, \dots) will be used to represent Lambek atoms, small Greek letters (α, β, \dots) will be used to represent formulae in the Lambek Calculus and capital Greek letters (Γ, Δ, \dots) will be used to represent sequences of formulae in the Lambek Calculus. The letter X will be used to represent a variable over the Lambek atoms.

When brackets are not necessary to distinguish meaning, they will be dropped. In addition, the sequence of formulae to the left of the turnstile will not be comma separated but simply shown as a list.

Figure 2.1: Sequents.

$$\begin{array}{l} (A/A) \ A \vdash ((A/A) \bullet A) \\ A \ (A/A) \ (((A/A)/A)/A) \vdash A \end{array}$$

Figures 2.2 and 2.3 give a sequent calculus presentation of the the Lambek calculus.

Figure 2.2: Axioms in the Lambek Calculus.

$$X \vdash X$$

Definition 2.1.4. A sequent $\Gamma \vdash \alpha$ is *derivable in L^\bullet* if there exists a proof from axioms in this logic.

Definition 2.1.5. A sequent $\Gamma \vdash \alpha$ is *derivable in $L^{\bullet*}$* if there exists a proof from axioms in this logic and in the rules $\backslash R$ and $/R$, Γ is non-empty.

Definition 2.1.6. A sequent $\Gamma \vdash \alpha$ is *derivable in L* if there exists a proof from axioms in this logic which uses neither $\bullet R$ nor $\bullet L$.

Figure 2.3: Inference rules in the Lambek Calculus.

$$\begin{array}{c}
\frac{\Gamma \vdash \alpha \quad \Delta \beta \Theta \vdash \gamma}{\Delta \Gamma \alpha \backslash \beta \Theta \vdash \gamma} \backslash L \qquad \frac{\alpha \Gamma \vdash \beta}{\Gamma \vdash \alpha \backslash \beta} \backslash R \\
\\
\frac{\Gamma \vdash \alpha \quad \Delta \beta \Theta \vdash \gamma}{\Delta \beta / \alpha \Gamma \Theta \vdash \gamma} / L \qquad \frac{\Gamma \alpha \vdash \beta}{\Gamma \vdash \beta / \alpha} / R \\
\\
\frac{\Gamma \alpha \beta \Delta \vdash \gamma}{\Gamma \alpha \bullet \beta \Delta \vdash \gamma} \bullet L \qquad \frac{\Gamma \vdash \alpha \quad \Delta \vdash \beta}{\Gamma \Delta \vdash \alpha \bullet \beta} \bullet R
\end{array}$$

Definition 2.1.7. A sequent $\Gamma \vdash \alpha$ is *derivable in L^** if there exists a proof from axioms in this logic which uses neither $\bullet R$ nor $\bullet L$ and in the rules $\backslash R$ and $/R$, Γ is non-empty.

Figure 2.4: Proof that the sequent $(A/A) \ A \vdash ((A/A) \bullet A)$ is in L^* .

$$\frac{A \vdash A \quad \frac{A \vdash A}{\vdash (A/A)} / R \quad A \vdash A}{A \vdash A \quad \frac{A \vdash A}{A \vdash ((A/A) \bullet A)} \bullet R} / L$$

Figure 2.5: Proof that the sequent $((A/(A \backslash A))/A) \ A \ (A \backslash A) \ (A \backslash A) \vdash A$ is in L^* .

$$\frac{A \vdash A \quad \frac{A \vdash A \quad A \vdash A}{A \ (A \backslash A) \vdash A} \backslash L}{(A \backslash A) \vdash (A \backslash A)} \backslash R \quad \frac{A \vdash A \quad A \vdash A}{A \ (A \backslash A) \vdash A} \backslash L}{((A/(A \backslash A))/A) \ A \ (A \backslash A) \ (A \backslash A) \vdash A} / L$$

Figures 2.4 and 2.5 show proofs in the Lambek Calculus. Note that if a sequent $\Gamma \vdash A$ contains the connective \bullet anywhere, it is not possible for $\Gamma \vdash A$ to be in either L or L^* .

Figure 2.6: The *CUT* rule.

$$\frac{\Gamma \vdash \alpha \quad \Delta\alpha\Theta \vdash \beta}{\Delta\Gamma\Theta \vdash \beta} (CUT)$$

2.1.2 The Lambek Calculus and *CUT*

The original formulation of the calculus in Lambek (1958) included the *CUT* rule in addition to those shown in figure 2.3. The cut rule is shown in figure 2.6.

Lambek (1958) proved the cut elimination theorem for the Lambek calculus and as such the *CUT* rule is only used when it simplifies proofs.

2.2 Complexity results for the Lambek Calculus

With our goal of determining the computational complexity of sequent derivability in the Lambek Calculus, we turn our eyes towards previous results. The complexity of derivability in two of our logics is known and for the other two, it is not. Pentus (2003) proved that derivability in L^\bullet and $L^{\bullet*}$ is *NP*-complete. As Penn (2001) has remarked, this result will not be easily extendable to L and L^* because the reduction relies very heavily on the use of products and a replacement in the product free fragment has not been forthcoming. We will refine the answer to this question in chapter 5.

In addition to these results, a number of variants of the Lambek Calculus have been shown to have polynomial time algorithms, as follows:

- L and L^* restricted to second-order formulae (Aarts (1994)).
- Non-associative L and L^* (Aarts and Trautwein (1994)).
- Non-associative L^\bullet and $L^{\bullet*}$ (de Groote (1999b)).

The complexity of parsing in L and L^* is currently unknown. This gives the impression that product may be the important difference in the tractability of the Lambek Calculus.

A major goal of this thesis is to try to isolate the intractable parts of parsing with the hope of restricting the calculus in various ways to achieve tractability.

Chapter 3

Proof nets for the Lambek Calculus

3.1 Proof nets

Girard (1987) introduced linear logic as a sub-structural logic, as expressively powerful as more traditional logics such as classical logic or intuitionistic logic but able to be separated into several self-contained fragments. It is well known that the *multiplicative fragment* of linear logic is equivalent to L^\bullet through a set of DeMorgan style laws (see Roorda (1991), Lamarche and Retore (1996), Pentus (1998), Moot (2004)).

In addition to Linear Logic, Girard (1987) introduced an extra-logical proof system which he named *proof nets*. The primary advantage of proof nets over proofs is that several proofs which are essentially the same share a single proof net, thus reducing spurious ambiguity. In this spirit, Girard (1987) proved that there is a bijection between proof nets and proofs in a certain normal form which can be used to easily map instances of proof nets to their corresponding proofs and vice versa. Much of the work already done on proof nets has been to posit new definitions and representations and to then prove a bijection between the new definition of proof nets and proofs in the Lambek Calculus with the aim of either reducing the computational complexity of proof net correctness or simplifying their presentation. A theorem proving this equivalence will be referred to as

a *proof net equivalence theorem*.

Proof structures are structures built upon a sequent in the calculus. Proof structures that satisfy certain conditions qualify as proof nets and thus correspond to proofs. The work done on proof nets can be neatly categorized based on the conditions which proof structures must satisfy to qualify as proof nets. There are two main sets of conditions: The Girard style correctness conditions, based on the original conditions of Girard (1987) (discussed in section 3.1.2) and the Roorda style correctness conditions, based on conditions originating in Roorda (1991) (discussed in section 3.1.3). We now proceed with definitions that are common to both styles of correctness conditions.

3.1.1 Proof structures

A proof structure is made up of two separate pieces: A deterministic forest called the proof frame and a non-deterministic bijection called the axiomatic linkage.

The proof frame

Our definition of the proof frame will be recursive and we first introduce the base case. The mapping of the Lambek Calculus into a subset of linear logic requires that each formula be assigned a polarity. We assign negative polarity (respectively positive polarity) denoted $-$ (respectively $+$) to the antecedents (respectively the succedent) of a sequent. These formulae will be referred to as the *terminal formulae*¹. Figure 3.1 shows the terminal formulae for a sequent.

Figure 3.1: Terminal formulae for the sequent $\alpha_1, \dots, \alpha_n \vdash \alpha$.

$$\alpha_1^- \dots \alpha_n^- \alpha^+$$

¹Other presentations of proof nets refer to these formulae as conclusions. This choice of terminology follows Penn (2001) to avoid confusion with conclusions of inference rules. In addition, some presentations of the proof frame use \circ for $+$ and \bullet for $-$.

We now continue with the recursive definition. We will break down each of the terminal formulae using rules similar to logical rules. These rules will either be referred to as *links* or as *proof frame building rules* depending on the context. The rules will build tree structures upon each of the terminal formulae:

Figure 3.2: Proof frame building rules.

$$\begin{array}{ccc}
 \frac{\alpha^+ \quad \beta^-}{(\alpha \setminus \beta)^-} \otimes & \frac{\alpha^- \quad \beta^+}{(\alpha / \beta)^-} \otimes & \frac{\alpha^- \quad \beta^-}{(\alpha \bullet \beta)^-} \wp \\
 \frac{\beta^+ \quad \alpha^-}{(\alpha / \beta)^+} \wp & \frac{\beta^- \quad \alpha^+}{(\alpha / \beta)^+} \wp & \frac{\beta^+ \quad \alpha^+}{(\alpha \bullet \beta)^+} \otimes
 \end{array}$$

At times, these rules will be grouped according to their connectives and polarities and other times, they will be grouped according to whether they are a \wp or \otimes link. The negative rules are those appearing in the top row of figure 3.2 while the positive rules are those appearing in the bottom row. The system of grouping rules by polarity and connectives will be used primarily when the system we are discussing deals exclusively with the Lambek Calculus. The \wp and \otimes rule groupings will be used for systems which were originally formulated for fragments of linear logic.

The proof frame building process is deterministic due to the fact that each connective-polarity pair has exactly one rule. Each of these six rules will be referred to as *links* in the context of a proof frame. Also, because each of these rules eliminates exactly one connective and preserves its two arguments, the formulae at the top of the proof frame are necessarily Lambek atoms. These atoms will be referred to as *axiomatic formulae*. Figures 3.3 and 3.4 show examples of proof frames.

In a proof frame, it commonly happens that the same Lambek formula appears in various distinct positions. It is often important then to distinguish between two occurrences of the same formula in different positions of a proof frame. When this difference is important, we will refer to these as two distinct *formula occurrences*. This notion can easily be formalized if necessary.

Figure 3.3: The proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$.

$$\frac{A^- \quad A^+}{(A/A)^-} \otimes \frac{A^+ \quad \frac{\frac{A^- \quad A^-}{(A \bullet A)^-} \wp \quad A^+}{(A/(A \bullet A))^+} \wp}{((A/(A \bullet A)) \bullet A)^+} \otimes$$

Figure 3.4: The proof frame for $(A/A) \quad A \vdash ((A/A) \bullet A)$.

$$\frac{A^- \quad A^+}{(A/A)^-} \otimes \quad A^- \quad \frac{A^+ \quad \frac{A^- \quad A^+}{(A/A)^+} \wp}{((A/A) \bullet A)^+} \otimes$$

Linking the axiomatic formulae

Once we have established the proof frame, we choose a bijection between the sets of positive and negative axiomatic formulae. Any bijection is allowed subject to the condition that the bijection only maps axiomatic formulae to axiomatic formulae consisting of identical Lambek atoms of opposite polarity. This bijection will be referred to as the *axiomatic linkage*. If the bijection maps an instance of X^+ to an instance of X^- , we say that this instance of X^+ and this instance of X^- are *linked*. Thus, an axiom link adds a seventh kind of link to the previous six. The axiomatic link is shown above the proof net as seen in figures 3.5, 3.6 and 3.7.

Figure 3.5: A proof structure for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$.

$$\frac{A^- \quad A^+}{(A/A)^-} \otimes \quad \frac{A^+ \quad \frac{\frac{A^- \quad A^-}{(A \bullet A)^-} \wp \quad A^+}{(A/(A \bullet A))^+} \wp}{((A/(A \bullet A)) \bullet A)^+} \otimes$$

A proof frame together with an axiomatic linkage will be called a proof structure. A proof structure that corresponds to a well-formed proof in the Lambek calculus is

Figure 3.6: A proof structure for $(A/A) \ A \vdash ((A/A) \bullet A)$.

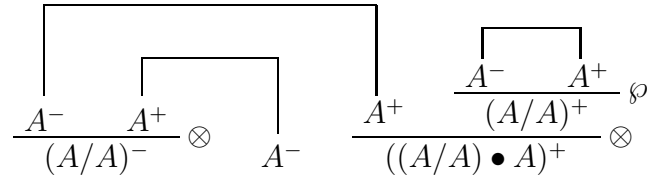
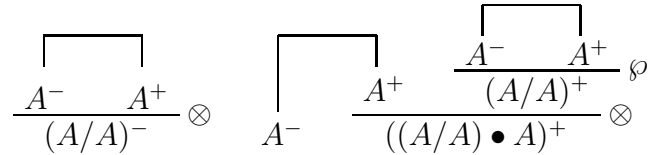


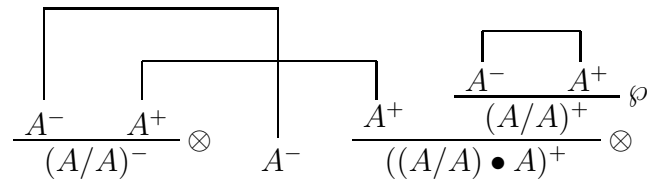
Figure 3.7: Another proof structure for $(A/A) \ A \vdash ((A/A) \bullet A)$.



called a proof net, although formal definitions of proof nets will follow. Note that not all proof structures correspond to well-formed proofs. It should be obvious to the reader that graphs are an ideal representation of proof structures and nearly all presentations of proof nets use this approach².

Our first correctness criterion is due to the fact that the Lambek calculus is non-commutative. For a proof structure to be a proof net its axiomatic linkage must be drawable as a planar graph in the half plane. We will not be formal here but a formal definition is possible if necessary. The proof structure in figure 3.8 has a non-planar linkage. It is important to note that the underlying sequent in a malformed proof structure may still be provable (because it has another proof structure that is well-formed).

Figure 3.8: A proof structure with a non-planar linkage for $(A/A) \ A \vdash ((A/A) \bullet A)$.



²Two notable exceptions are the presentations of Girard (1987) and that of Lamarche (1995), although both use structures which bear a strong resemblance to graphs. In addition, de Groote (1999a) provides correctness criteria based on algebraic constraints.

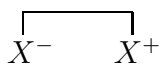
3.1.2 Girard style correctness

The proof structure correctness criteria schemes based on the original formulation by Girard (1987) all build an undirected graph out of the proof structure. The correctness criteria then take the form of a simple acyclicity condition. Many formulations have a connectedness condition but Fadda and Morrill (2005) show that these conditions are unnecessary. The one exception to the acyclicity requirement is Girard’s original formulation which requires the proof structure to have a “long trip” or Hamiltonian cycle but the idea is essentially the same (see Danos and Regnier (1989)).

Two presentations of the correctness criteria will be presented here but other presentations exist. Roorda (1991) introduces *Quantum graphs* (which are completely distinct from Roorda’s work presented in section 3.1.3) and Carpenter and Morrill (2005) introduces *Switch graphs* both of which provide similar functionality to both the work of Danos and Regnier (1989) and the work of Retore (1996).

All of the correctness criteria from this tradition treat the \wp links the same regardless of their polarity and the same holds true for \otimes links. In addition, the correctness criteria do not differentiate between the two possible orderings of premises for \otimes^- and \wp^+ . Thus, the original seven links are reduced to the following three for this section:

Table 3.1: Three links of proof structures for Girard style correctness.

Axiom	\wp	\otimes
	$\frac{\beta \quad \alpha}{\alpha \wp \beta}$	$\frac{\beta \quad \alpha}{\alpha \otimes \beta}$

The graph formalisms of this section will be specified according to their translation of these three links into vertices and edges.

Due primarily to the fact that the foundations of these methods lie in linear logic and not specifically in the Lambek calculus, the correctness criteria given here pertain only

to L and L^\bullet , not L^* and $L^{\bullet*}$. It is not difficult to add correctness criteria to account for L^* and $L^{\bullet*}$.

Danos and Regnier (1989)

Danos and Regnier (1989) were the first to explicitly formulate proof nets as graphs. Subsequent treatments of the topic have reversed the sets of vertices and edges yielding an equivalent but much more intuitive presentation (see Roorda (1991), Retore (1996), Carpenter and Morrill (2005)). We will refer to these graphs as *DR*-graphs. Obtaining the *DR*-graph from a proof structure requires the translation in table 3.2.

Table 3.2: Translation from proof structures to *DR*-graphs.

	Proof Structure	$G = \langle V, E \rangle$
Axiom	$\overline{X^- \quad X^+}$	$X^- \circ \text{---} \circ X^+$
\wp	$\frac{\alpha \quad \beta}{\alpha \wp \beta}$	$\alpha \circ \text{---} \circ \beta$ $\alpha \wp \beta$
\otimes	$\frac{\alpha \quad \beta}{\alpha \otimes \beta}$	$\alpha \circ \text{---} \circ \beta$ $\alpha \otimes \beta$

We are now in a position to define the correctness criteria.

Definition 3.1.1. A *switching* of a *DR*-graph is a *DR*-graph with one of each pair of edges introduced by a \wp link deleted.

Definition 3.1.2. A proof structure is a proof net if and only if every switching of the graph is acyclic³.

³Fadda and Morrill (2005) show that connectedness of switchings is implied by the acyclicity of switching and the fact that the calculus is intuitionistic.

Then, Danos and Regnier (1989) proved the proof net equivalence theorem for *DR*-graphs.

Of the three *DR*-graphs shown in figures 3.9, 3.10 and 3.11, only the proof structure corresponding to the graph in figure 3.10 is a proof net.

Figure 3.9: The *DR*-graph for the proof structure in figure 3.5.

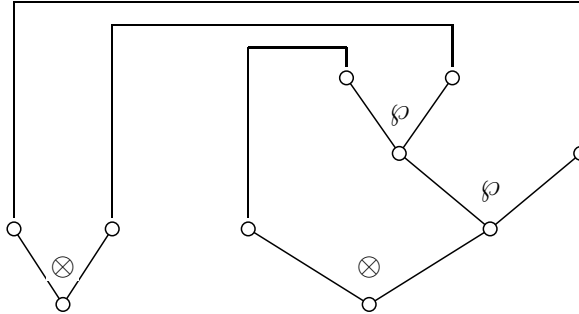


Figure 3.10: The *DR*-graph for the proof structure in figure 3.6.

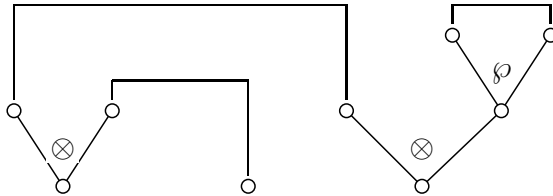
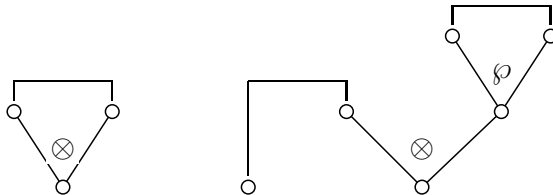


Figure 3.11: The *DR*-graph for the proof structure in figure 3.7.

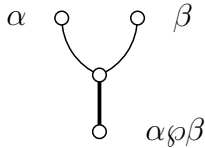
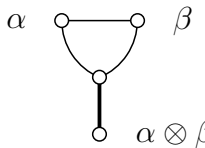


R & B graphs

It is apparent that the presentation above lacks elegance (in requiring a definition like *switching*) and is potentially computationally expensive (in requiring the checking of an exponential number of graphs to check correctness of a single proof structure). Ror-
 orda (1991) and Carpenter and Morrill (2005) both give presentations to remedy these
 problems but here the presentation of Retore (1996) will be given for its simplicity and
 tractability.

R & B graphs as introduced by Retore (1996) introduce a graph similar to a *DR*-
 graph except that each edge is identified as either bold or regular⁴. *B* is the set of bold
 edges and *R* is the set of regular edges where *B* and *R* form a partition of *E* and *B* is a
 perfect matching. We get the following translation:

Table 3.3: Translation from proof structures to *R&B*-graphs.

	Proof Structure	$G = \langle V, E \rangle$
Axiom	$\overline{X^- \quad X^+}$	$X^- \text{ --- } X^+$
\wp	$\frac{\alpha \quad \beta}{\alpha \wp \beta}$	
\otimes	$\frac{\alpha \quad \beta}{\alpha \otimes \beta}$	

Definition 3.1.3. In an *R&B*-graph, an \ae -cycle is an even length cycle whose edges are
 alternately in *B* and in *R*.

Definition 3.1.4. A proof structure is a proof net if and only if the *R&B* graph contains

⁴Retore (1996) also refers to bold as blue and regular as red.

no \ae -cycles.

A simple proof in addition to a result in Fadda and Morrill (2005) shows that \ae -acyclicity is enough to prove correctness.

Then, Retore (1996) proved the proof net equivalence theorem for $R\&B$ graphs. In addition, Retore (1999) proved that the question of whether a proof structure is a proof net can be checked in $O(n^2)$ time by checking whether its $R\&B$ -graph is \ae -acyclic.

Like the examples of DR -graphs, of the three $R\&B$ -graphs in figures 3.12, 3.13 and 3.14, only the $R\&B$ -graph in figure 3.13 is \ae -acyclic.

Figure 3.12: The $R\&B$ -graph for the proof structure in figure 3.5.

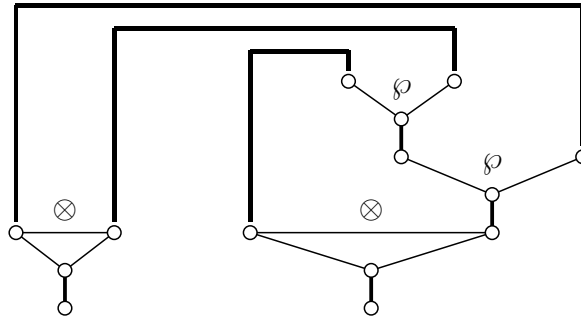
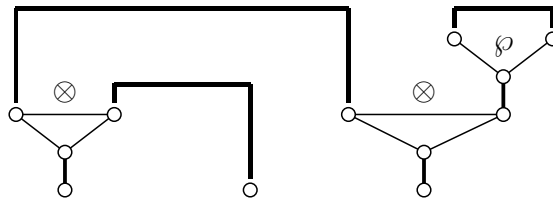


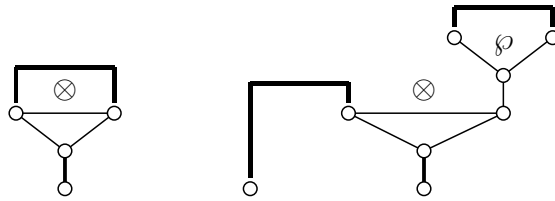
Figure 3.13: The $R\&B$ -graph for the proof structure in figure 3.6.



3.1.3 Roorda style correctness

Roorda (1991) first introduced a set of correctness criteria for proof nets of Lambek calculus sequents that were markedly different from the original criteria of Girard (1987).

There are several key differences:

Figure 3.14: The $R&B$ -graph for the proof structure in figure 3.7.

- They require the view of a proof net as a *directed* graph.
- They differentiate between the positive and negative occurrences of proof frame links.
- They differentiate between the order of the premises of proof frame links.
- They require a special path accessibility condition based on the positive / and \ link.

Otherwise, the conditions still involve acyclicity and connectedness but there is little correspondence between the cycles for graphs defined in the Roorda tradition and the cycles for graphs defined in the Girard tradition. This can be seen by comparing figures 3.5, 3.6 and 3.7 with figures 3.18, 3.19 and 3.20. The first set of figures shows the DR -graphs of three proof structures while the second set shows the variable substitutions of the same three proof structures.

Lamarche (1995) introduces a similar set of conditions for proof nets of sequents in linear logic in its full form. Because we are only interested in the Lambek Calculus in this thesis, we will use the formulation by Roorda (1991). Also, the presentation given in Roorda (1991) and the graphs introduced by Penn (2001) provide a simpler graph than the graphs presented in Lamarche (1995).

Like the original formulation by Girard, Roorda's original formulation did not involve graphs at all and the first explicit formulation of these conditions as graph conditions

comes from Penn (2001). Furthermore, Penn (2001) only defined such graphs for L and L^* and as such, the only graph-based correctness conditions which will be presented in this section are for these calculi.

Roorda's Correctness Criteria

Roorda's correctness criteria require the annotation of the proof frame with lambda calculus terms in the following way.

We begin with a countably infinite set of variables V . A *fresh* variable of V is one which has not been used previously. Then, each of the terminal formulae is annotated with a fresh lambda variable. The remaining formulae in the proof frame receive terms according to the following rules (which operate in parallel to the proof frame building rules):

Figure 3.15: Rules for annotating proof frames with lambda terms.

$$\begin{array}{c}
 \frac{\alpha \dagger u \quad \beta \bar{\cdot} tu}{\alpha \backslash \beta \bar{\cdot} t} \quad \frac{\alpha \bar{\cdot} tu \quad \beta \dagger u}{\alpha / \beta \bar{\cdot} t} \quad \frac{\alpha \bar{\cdot} (t)_0 \quad \beta \bar{\cdot} (t)_1}{\alpha \bullet \beta \bar{\cdot} t} \\
 \frac{\beta \dagger v' \quad \alpha \bar{\cdot} u}{\alpha \backslash \beta \dagger v} \quad \frac{\beta \bar{\cdot} u \quad \alpha \dagger v'}{\alpha / \beta \dagger v} \quad \frac{\beta \dagger v' \quad \alpha \dagger v''}{\alpha \bullet \beta \dagger v}
 \end{array}$$

It can easily be seen that every formula in the proof frame is assigned a lambda term in this way. In addition, notice that formulae with positive polarity are always labeled with lambda variables whereas formulae with negative polarity can have more complex lambda terms. Figures 3.16 and 3.17 show two annotated proof frames.

Next, we build a set of substitutions in the following way. We begin with an empty set and for each of the positive rules, we add one substitution to the set as shown in table 3.4.

Then, for each pair of linked axiomatic formulae $X^+ : v$ and $X^- : t$, we add the substitution $[v := t]$. The set of substitutions for a proof structure can be seen in figures

Figure 3.16: Lambda terms annotated to the proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$.

$$\frac{\frac{A \bar{\vdash} ab \quad A \dagger \vdash b}{(A/A) \bar{\vdash} a} \otimes \quad \frac{A \dagger \vdash d}{((A/(A \bullet A)) \bullet A) \dagger \vdash c} \otimes \quad \frac{\frac{\frac{A \bar{\vdash} (f)_0 \quad A \bar{\vdash} (f)_1}{(A \bullet A) \bar{\vdash} f} \wp \quad A \dagger \vdash g}{(A/(A \bullet A)) \dagger \vdash e} \wp}{((A/(A \bullet A)) \bullet A) \dagger \vdash c} \otimes$$

Figure 3.17: Lambda terms annotated to the proof frame for $(A/A) \vdash A \vdash ((A/A) \bullet A)$.

$$\frac{\frac{A \bar{\vdash} ab \quad A \dagger \vdash b}{(A/A) \bar{\vdash} a} \otimes \quad A \bar{\vdash} c \quad \frac{\frac{A \bar{\vdash} g \quad A \dagger \vdash h}{(A/A) \dagger \vdash f} \wp \quad A \dagger \vdash e}{((A/A) \bullet A) \dagger \vdash d} \otimes$$

3.18, 3.19 and 3.20.

We begin with the variable labeling the unique positive terminal formula and expand with the variable substitutions until no more expansions are possible. It is this process that the correctness criteria are defined upon.

Definition 3.1.5. A proof structure is a proof net if and only if the following are satisfied:

- **PN(1)** There is precisely one terminal formula with positive polarity.
- **PN(2)** Variable expansion terminates.

If the above conditions hold for a proof net, then we can discuss the term t generated by variable substitution beginning with the label of the positive terminal formula.

- **PN(3)** If t contains a subterm $\lambda v.s$ then v occurs in s and does not occur outside s .
- **PN(4)** Every variable assigned to a negative terminal formula occurs in t .

To extend these correctness criteria to L^* and $L^{\bullet*}$, we must add one more condition:

Table 3.4: Substitutions due to positive polarity rules.

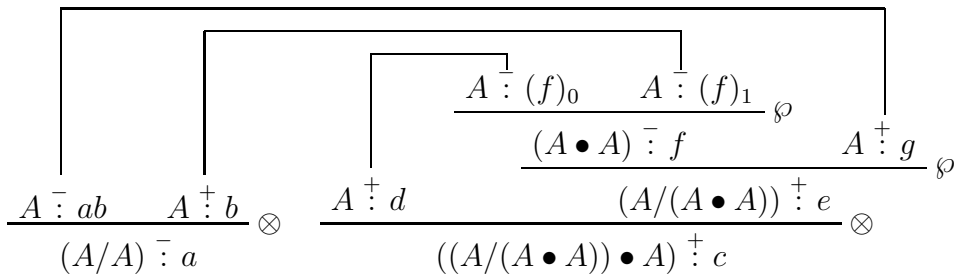
Rule	Substitution
$\frac{\beta \dagger v' \quad \alpha \bar{u}}{\alpha \setminus \beta \dagger v}$	$[v := \lambda u.v']$
$\frac{\beta \bar{u} \quad \alpha \dagger v'}{\alpha / \beta \dagger v}$	$[v := \lambda u.v']$
$\frac{\beta \dagger v' \quad \alpha \dagger v''}{\alpha \bullet \beta \dagger v}$	$[v := \langle v'', v' \rangle]$

- **PN(CT)** t has no closed subterms.

Then, Roorda (1991) proved the proof net equivalence theorem for this system.

The proof structure shown in figure 3.18 does not qualify as a proof net because it does not satisfy $PN(3)$, the proof structure in figure 3.19 does qualify as a proof net, and the proof structure in figure 3.20 does not qualify because it violates $PN(4)$.

These correctness criteria are unwieldy and difficult to check and what follows is a more familiar graph presentation.

 Figure 3.18: Variable substitution of a proof structure for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$.


Substitutions := $[c := \langle e, d \rangle], [e := \lambda f.g], [g := ab], [b := (f)_1], [d := (f)_0]$

Variable Substitution := $c \rightarrow \langle e, d \rangle \rightarrow \langle \lambda f.g, (f)_0 \rangle \rightarrow \langle \lambda f.ab, (f)_0 \rangle \rightarrow \langle \lambda f.a(f)_1, (f)_0 \rangle$

Figure 3.19: Variable substitution of a proof structure for $(A/A) \vdash A \vdash ((A/A) \bullet A)$.

$$\begin{array}{c}
\begin{array}{c} \overline{A \vdash ab} \quad \overline{A \dagger b} \\ \otimes \\ \overline{(A/A) \vdash a} \end{array} \quad \begin{array}{c} \overline{A \vdash c} \\ \otimes \\ \overline{(A/A) \bullet A \dagger d} \end{array} \\
\hline
\overline{A \dagger e} \quad \frac{\overline{A \vdash g} \quad \overline{A \dagger h}}{\overline{(A/A) \dagger f}} \wp
\end{array}$$

Substitutions := $[d := \langle f, e \rangle], [f := \lambda g.h], [e := ab], [b := c], [h := g]$

Variable Substitution := $d \rightarrow \langle f, e \rangle \rightarrow \langle \lambda g.h, ab \rangle \rightarrow \langle \lambda g.g, ac \rangle$

Figure 3.20: Variable substitution of another proof structure for $(A/A) \vdash A \vdash ((A/A) \bullet A)$.

$$\begin{array}{c}
\begin{array}{c} \overline{A \vdash ab} \quad \overline{A \dagger b} \\ \otimes \\ \overline{(A/A) \vdash a} \end{array} \quad \begin{array}{c} \overline{A \vdash c} \\ \otimes \\ \overline{(A/A) \bullet A \dagger d} \end{array} \\
\hline
\overline{A \dagger e} \quad \frac{\overline{A \vdash g} \quad \overline{A \dagger h}}{\overline{(A/A) \dagger f}} \wp
\end{array}$$

Substitutions := $[d := \langle f, e \rangle], [f := \lambda g.h], [b := ab], [e := c], [h := g]$

Variable Substitution := $d \rightarrow \langle f, e \rangle \rightarrow \langle \lambda g.h, c \rangle \rightarrow \langle \lambda g.g, c \rangle$

LC-Graphs

Penn (2001) introduced a graph representation of the substitutions of Roorda (1991) to clarify the exact computation necessary. Penn only provided graph representations for L and L^* and not the calculi with product.

Definition 3.1.6. An LC-Graph $G = \langle V, E \rangle$ is defined as follows:

- V is the set of variables occurring as annotations to the proof frame.
- $(u, v) \in E$ if and only if u occurs on the left side of a substitution and v occurs on the right side.

We must have a few definitions before we can state the correctness criteria.

Definition 3.1.7. For a substitution $[v := \lambda u.v']$, we call $v \in V$ a lambda node, $v' \in V$ its plus daughter and $u \in V$ its minus daughter.

Now, we can proceed with the correctness criteria.

Definition 3.1.8. A proof structure is a proof net if and only if its LC-Graph satisfies the following conditions:

- **I(1)** There is a unique node in G with in-degree 0, from which all other nodes are path accessible.
- **I(2)** G is acyclic.
- **I(3)** For every lambda node $v \in V$, there is a path from its plus daughter u to its minus daughter w .

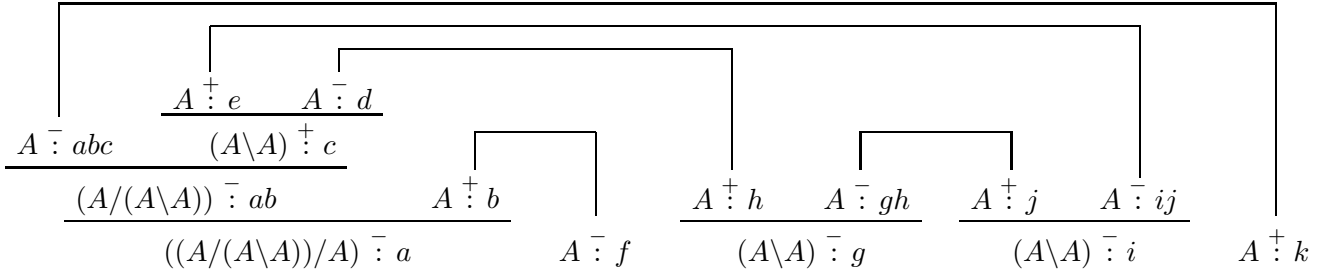
To extend these correctness criteria to L^* , we must add one more condition:

- **I(CT)** For every lambda node $v \in V$, there is a path in G , $v \rightsquigarrow x$, where x is a terminal node and there is no lambda node $v' \in V$ such that $v \rightsquigarrow v' \rightarrow x$.

Then, Penn (2001) proved the proof net equivalence theorem for LC graphs.

Figures 3.21 and 3.23 show proof structures and variable substitutions for $((A/(A \setminus A))/A) \ A \ (A \setminus A) \ (A \setminus A) \vdash A$ (taken from Penn (2001)). Figures 3.22 and 3.24 show the LC graphs for these proof structures.

Figure 3.21: A proof structure for $((A/(A\backslash A))/A) \ A \ (A\backslash A) \ (A\backslash A) \vdash A$.



Substitutions := $[c := \lambda d.e], [k := abc], [e := ij], [h := d], [b := f], [j := gh]$

Variable Substitution := $k \rightarrow abc \rightarrow af\lambda d.e \rightarrow af\lambda d.ij \rightarrow af\lambda d.igh \rightarrow af\lambda d.igd$

Figure 3.22: The LC graph for the proof structure in figure 3.21.

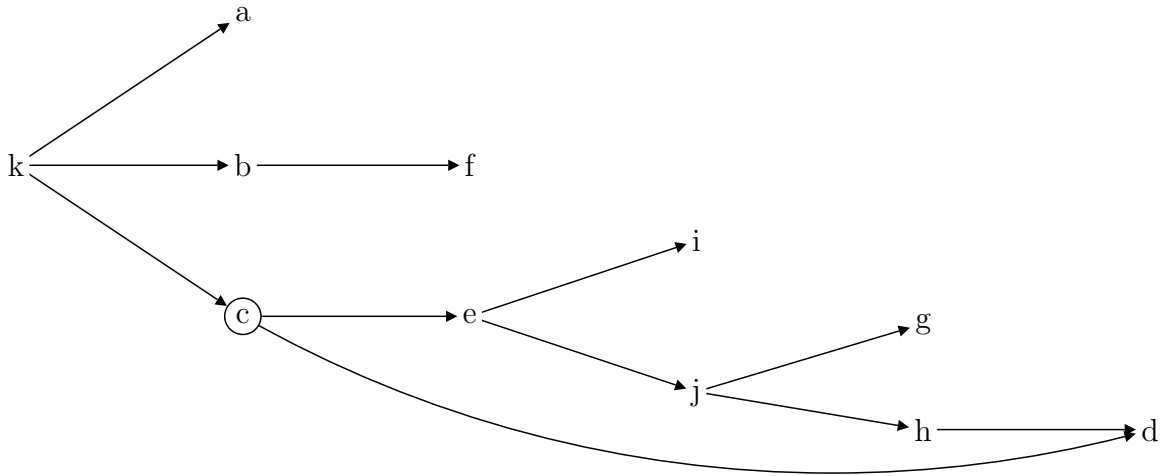
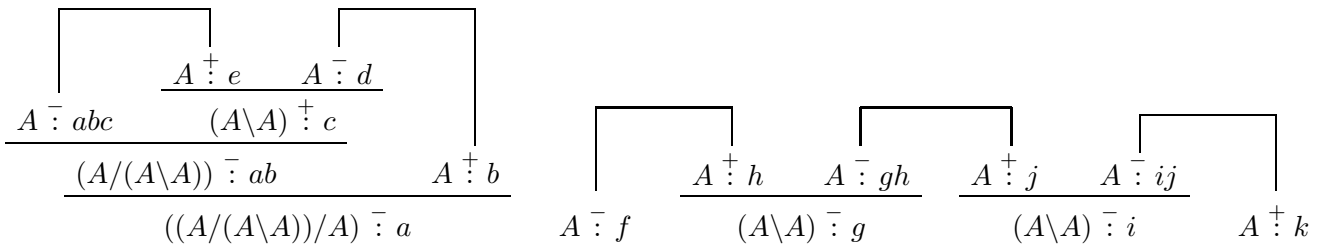
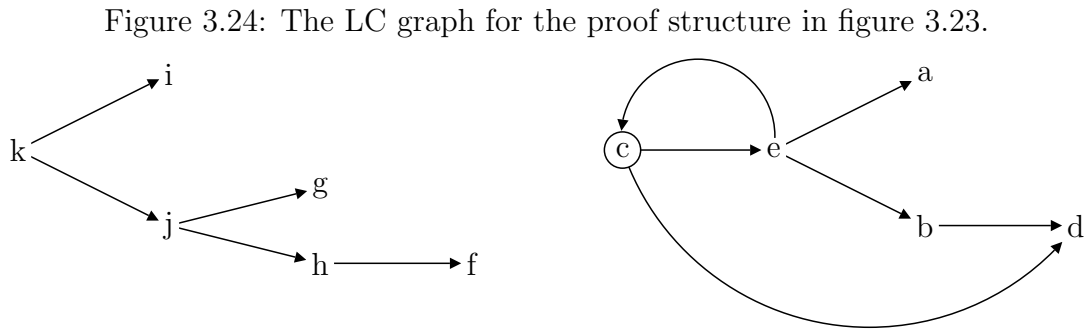


Figure 3.23: Another proof structure for $((A/(A\backslash A))/A) \ A \ (A\backslash A) \ (A\backslash A) \vdash A$.



Substitutions := $[c := \lambda d.e], [k := ij], [j := gh], [h := f], [b := d], [e := abc]$

Variable Substitution := $k \rightarrow ij \rightarrow igh \rightarrow igf$



Chapter 4

LC Graphs for L^\bullet

This section extends the notion of LC graph from L to L^\bullet . In this spirit, additional structure is put into place to handle products and conditions are provided for determining correctness. Much of this section follows the structure of Penn (2001). Some of the proofs provided here are similar to those in Penn (2001). Where the proofs would be identical, the reader is referred to that article.

4.1 Some additional terminology for proof nets

Definition 4.1.1. A *child* of a non-axiomatic formula occurrence x in a proof frame is either of the two formulae occurrences that are premises of the rule used to decompose x . Similarly, the *parent* of a non-terminal formula occurrence y is the formula occurrence that is the conclusion of the rule in which y is a premise.

Definition 4.1.2. The *ancestor* relation is the transitive closure of the parent relation. The *descendant* relation is the transitive closure of the child relation.

Definition 4.1.3. A *positive (negative) path descendant* x of a positive (negative) formula occurrence y is a descendant of y such that x , y and all formula occurrences which are descendants of y and ancestors of x are positive (negative).

Figure 4.1: The proof frame for $(A/A) \vdash ((A/(A \bullet A)) \bullet A)$.

$$\begin{array}{c}
\frac{A \bar{\vdash} (f)_0 \quad A \bar{\vdash} (f)_1}{(A \bullet A) \bar{\vdash} f} \quad A \dagger \vdash g \\
\frac{A \bar{\vdash} ab \quad A \dagger \vdash b \quad A \dagger \vdash d \quad \frac{(A \bullet A) \bar{\vdash} f \quad A \dagger \vdash g}{(A/(A \bullet A)) \dagger \vdash e}}{(A/A) \bar{\vdash} a \quad ((A/(A \bullet A)) \bullet A) \dagger \vdash c} \\
\text{Substitutions := } [c := \langle e, d \rangle], [e := \lambda f.g]
\end{array}$$

In figure 4.1, $(A/(A \bullet A)) \dagger \vdash e$ is the parent of $(A \bullet A) \bar{\vdash} f$ and $A \bar{\vdash} ab$ is the child of $(A/A) \bar{\vdash} a$. $A \dagger \vdash g$ and $(A/(A \bullet A)) \dagger \vdash e$ are positive path descendants of $((A/(A \bullet A)) \bullet A) \dagger \vdash c$ and $A \bar{\vdash} (f)_0$ is a negative path descendant of $(A \bullet A) \bar{\vdash} f$.

4.2 LC Graphs

LC graphs as introduced by Penn (2001) were presented in section 3.1.3. We now introduce the definitions necessary to extend these to L^\bullet .

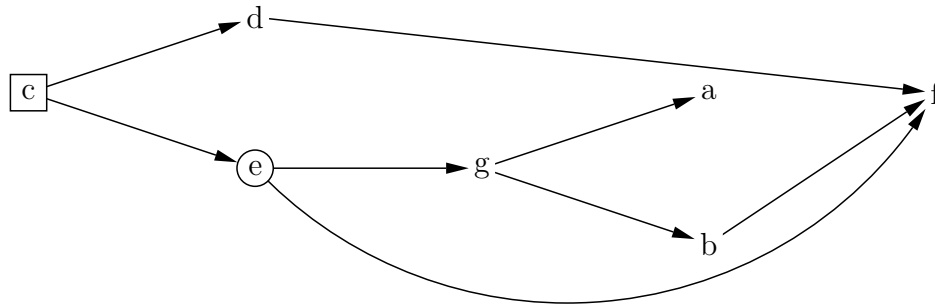
Definition 4.2.1. In a proof frame, a variable v that labels any formula $X \dagger \vdash v$ is called a *plus variable*. If X is a basic category, we distinguish v by saying it is a *simple plus variable*. If X is a complex category whose largest scoping connective is either \backslash or $/$, we distinguish v by saying it is a *lambda variable*. If X is a complex category whose largest scoping connective is \bullet , we distinguish v by saying it is a *paired variable*. A variable v that labels a formula $X \bar{\vdash} v$ is a *minus variable*.

Example 4.2.2. In figure 4.1, b , d and g are simple plus variables, e is a lambda variable, c is a paired variable and a and f are minus variables.

Figure 4.2 shows the LC graph for the proof structure shown in figure 3.18.

Proposition 4.2.3. *The simple plus variables, lambda variables, paired variables and minus variables partition the set of Lambek atoms in a proof frame.*

Figure 4.2: The LC Graph for the proof structure in 3.18.



Proof. Easily proved by examining the proof frame rules in figure 3.15 and the definitions of the variables. \square

Definition 4.2.4. The LC-Graph of a proof structure $G = \langle V, E \rangle$ is defined as follows:

- V is the set of lambda variables occurring as annotations to the proof frame.
- $(u, v) \in E$ if and only if u occurs on the left side of a substitution and v occurs on the right side.

It should be noted that our definition of LC graphs on product free sequents does not change to handle sequents with product. The difference in the graphs comes from the definitions of substitutions with product.

To distinguish lambda nodes and paired nodes, lambda nodes are circled in graphical representations of LC graphs and paired nodes are drawn as boxes. Note that f has an in-degree 3 in figure 4.2 because it is the minus daughter of a lambda node and because it appears in 2 projections.

Because of the definition of V , we will use the terms “vertices” (in an LC-graph) and “variables” (in a proof frame) interchangeably.

4.3 Basic Properties of LC Graphs

Proposition 4.3.1. *Let t be a term labeling a negative formula in a proof frame. Then, t contains exactly one minus variable.*

Proof. The result follows from an easy application of induction to the proof frame building rules of figure 3.15. \square

Proposition 4.3.2. *Let x be a negative formula occurrence. The label of any negative path descendant of x has the label of x as a subterm. In addition, there exists a negative path descendant of x that is an axiomatic formula.*

Proof. By examining the three proof frame building rules in figure 3.15, the following two facts can be observed:

- All three negative proof frame rules have at least one negative premise
- For every negative proof frame rule, each negative premise contains the label of the conclusion as a subterm

Then, by a simple application of induction, the result is proved. \square

Proposition 4.3.3. *In an LC graph, the only plus node with an in-degree of 0 is the label of the positive terminal formula.*

Proof. Let p be a plus node that does not label a terminal formula. In the proof frame, such a label is introduced in one of three ways:

1. As the result of a negative occurrence of a \setminus or $/$ rule.

In this case, p labels the positive premise and the label of the negative premise includes p . Then, by proposition 4.3.2, there exists some axiomatic formula which contains p . Since every negative axiomatic formula occurs at one end of an axiom link, p has an in-degree of at least 1.

2. As the result of a positive occurrence of a \setminus or $/$ rule.

In this case, p labels the plus daughter of a lambda node. By the definition of LC Graphs, there will be an edge from the lambda node to p . Therefore, p has an in-degree of at least 1.

3. As the result of a positive occurrence of a \bullet rule.

In this case, p labels one of the plus daughters of a paired node. By the definition of LC Graphs, there will be an edge from the paired node to p . Therefore, p has an in-degree of at least 1.

Therefore, any plus node which is not a label of a positive terminal formula must have an in-degree of at least 1. \square

Proposition 4.3.4. *Minus nodes have out-degree 0.*

Proof. Minus nodes cannot occur on the left hand side of a substitution by the definition of substitution, yielding minus nodes with out-degree 0. \square

Proposition 4.3.5. *In an LC graph, every plus node has an out-degree of at least 1. In other words, there are no terminal plus nodes.*

Proof. Proved in Penn (2001). \square

4.4 Unitary

In this section, our aim will be to provide a definition that is equivalent to the definition of “counts for one” in Roorda (1991) but which is more elegant, easier to work with and more precise. “Counts for one” is imprecise in part because Roorda’s original definition does not match what “counts for one” intuitively represents as shown by his examples.

4.4.1 Definitions

We begin with a definition of the λ -calculus with projections and pairings. This is adapted from Barendregt (1981).

Definition 4.4.1. Let V be a countably infinite set of variables. The set of terms Λ over V is recursively defined as follows:

- $v \in \Lambda$ for $v \in V$,
- $(\lambda x.r) \in \Lambda$ for $x \in V$ and $r \in \Lambda$ (Abstraction),
- $\langle q, r \rangle \in \Lambda$ for $q, r \in \Lambda$ (Pairing),
- $(r)_0 \in \Lambda$ and $(r)_1 \in \Lambda$ for $r \in \Lambda$ (Projection), and
- $(qr) \in \Lambda$ for $q, r \in \Lambda$ (Application).

Latin letters near the end of the alphabet (q, r, s, t, \dots) will be used to represent terms. In an abstraction $(\lambda x.r)$, the variable x is referred to as the abstracted variable. We will drop the brackets on applications whenever the situation allows and assume that a sequence of applications associates to the right.

Definition 4.4.2. A term t will be called a variable, abstraction, pairing, projection or application depending on whether its topmost syntactic rule from above was respectively the variable, abstraction, pairing, projection or application rule.

Example 4.4.3. $\lambda x.(y\langle x, z \rangle)w$ is an application in Λ .

Example 4.4.4. $\langle (x)_0, (x)_1 \rangle$ is a pairing in Λ .

Definition 4.4.5. A multiset is a pair (A, f) where A is a set and f is a function such that $f : A \rightarrow \mathbb{N}$ indicating the multiplicity of the elements in A .

Definition 4.4.6. Given two multisets (A, f) and (B, g)

- $(A, f) \cup (B, g) = (C, h)$ where $C = A \cup B$ and $h(x) = f(x) + g(x)$ if $x \in A \cap B$, $h(x) = f(x)$ if $x \notin B$ and $h(x) = g(x)$ if $x \notin A$.
- $(A, f) \setminus (B, g) = (C, h)$ where $C = A \setminus B$ and $h(x) = \min(f(x) - g(x), 0)$ if $x \in B$ and $h(x) = f(x)$ otherwise.

When possible, we will dispense with the notation (A, f) and identify multisets with capital Latin letters M, N, \dots . We will present multisets as comma separated lists remembering that order within each list is arbitrary.

Definition 4.4.7. We define Ξ as the set of multisets of terms. In addition, let $\Upsilon \subset \Xi$ be such that for $M \in \Upsilon$, each abstraction occurring either in M or as a subset of a term in M has a unique abstracted variable.

We wish to define a notion of *alpha*-equivalence between multisets in Ξ . A formal definition from our definitions of multisets is possible but instead we present a more intuitive definition by viewing multisets as sets with the possibility that distinct elements can be identical.

Definition 4.4.8. $M, N \in \Xi$ are α -equivalent if and only if there is a bijection $\eta : M \rightarrow N$ such that for $m \in M$, m is α -equivalent to $\eta(m)$.

Example 4.4.9. $\{\lambda x.x, \lambda y.y\} \in \Upsilon$.

Example 4.4.10. $\{\lambda x.x, \lambda x.x\} \in \Xi$ but $\{\lambda x.x, \lambda x.x\} \notin \Upsilon$.

Example 4.4.11. $\{\lambda x.x, \lambda y.y\}$ and $\{\lambda x.x, \lambda x.x\}$ are α -equivalent.

Definition 4.4.12. Let $M, N \in \Upsilon$. Let \succ be defined as the smallest relation satisfying the following criteria:

- $M \succ N$ if $\lambda x.r \in M$ and $N = (M \setminus \{\lambda x.r\}) \cup \{r\}$.
- $M \succ N$ if $\langle q, r \rangle \in M$ and $N = (M \setminus \{\langle q, r \rangle\}) \cup \{q, r\}$.
- $M \succ N$ if $(r)_0, (r)_1 \in M$ and $N = (M \setminus \{(r)_0, (r)_1\}) \cup \{r\}$.
- $M \succ N$ if $(st) \in M$ and $N = (M \setminus \{(st)\}) \cup \{s, t\}$.

Example 4.4.13. $(x)_0(x)_1 \succ (x)_0, (x)_1 \succ x$

Example 4.4.14. $(u(x)_0)_0(u(x)_0)_1(x)_1 \succ (u(x)_0)_0, (u(x)_0)_1(x)_1 \succ (u(x)_0)_0, (u(x)_0)_1, (x)_1 \succ u(x)_0, (x)_1 \succ u, (x)_0, (x)_1 \succ u, x$

Example 4.4.15. $\lambda x.x \succ x$

Definition 4.4.16. \succ^* is the reflexive and transitive closure of \succ .

Proposition 4.4.17. *If $M \succ^* N$ and $M \succ^* P$, then there exists an N' such that $N \succ^* N'$ and $P \succ^* N'$.*

Proof. No \succ -redex is disturbed by \succ -reduction at a different redex. \square

Definition 4.4.18. $M \in \Upsilon$ is \succ -minimal if there is no $N \in \Upsilon$ such that $M \succ N$.

Corollary 4.4.19. *Let $M \in \Upsilon$ and let $N, P \in \Upsilon$ such that $M \succ^* N$ and $M \succ^* P$ and both N and P are \succ -minimal. Then $N = P$. That is, reduction under \succ^* satisfies the Church-Rosser property.*

Definition 4.4.20. Let $M \in \Upsilon$. A \succ -minimal $N \in \Upsilon$ must exist such that $M \succ^* N$ since any single step of \succ strictly reduces the number of abstractions, pairings, projections or applications. By the previous corollary, N is unique. We denote N by $\Phi(M)$.

Definition 4.4.21. Let $M \in \Upsilon$. Then, m is *unitary* in M if and only if m occurs in exactly one of the terms of $\Phi(M)$ and that term is the variable m .

Definition 4.4.22. $M \in \Upsilon$ is *unitary* if and only if for every variable m in M , m is unitary in M .

Definition 4.4.23. $M \in \Xi \setminus \Upsilon$ is *unitary* if and only if there exists an $N \in \Upsilon$ such that M and N are α -equivalent and N is unitary.

Definition 4.4.24. A term t is *unitary* if and only if $\{t\}$ is unitary.

Example 4.4.25. xyz is unitary.

Example 4.4.26. $(x)_0yz$ is not unitary but it is \succ -minimal.

Example 4.4.27. $\Phi((\lambda x.x\langle y, z \rangle w))$ is x, y, z, w .

Example 4.4.28. Let $M = \{\lambda x.x\lambda x.x\}$. Then, M is unitary because M is α -equivalent to $\lambda x.x\lambda y.y$ and $\Phi(\lambda x.x\lambda y.y) = x, y$.

4.4.2 Basic Theorems

Proposition 4.4.29. *Let $M \succ^* N$. Then, $\Phi(M) = \Phi(N)$.*

Proof. We know that $N \succ^* \Phi(N)$ by definition. But then, $M \succ^* N \succ^* \Phi(N)$ and since $\Phi(N)$ is \succ -minimal, $\Phi(N) = \Phi(M)$. \square

Corollary 4.4.30. *Let $M \succ^* N$. Then N is unitary if and only if M is unitary.*

Lemma 4.4.31. *Let $M \in \Upsilon$ such that $x \in M$ is a variable with multiplicity at least 2. Then, M is not unitary.*

Proof. Upon examination of \succ , it is clear that a variable cannot participate in any of the reductions. Therefore, the multiplicity of x in $\Phi(M)$ will be at least that of the multiplicity of x in M . Therefore, M is not unitary. \square

Lemma 4.4.32. *Let $M \in \Upsilon$ such that $t \in M$ is a term with multiplicity at least 2. Then, M is not unitary.*

Proof. By structural induction:

- Base Case: t is a variable

Proven by the previous lemma.

- Inductive Step:

$$- t = \lambda x.r$$

Then, $M \succ^* N$ where $N = (M \setminus \{t, t\}) \cup \{r, r\}$ and by the inductive hypothesis,

N is not unitary, so M is not unitary.

$$- t = \langle s, r \rangle$$

Then, $M \succ^* N$ where $N = (M \setminus \{t, t\}) \cup \{s, r, s, r\}$ and by the inductive hypothesis, N is not unitary, so M is not unitary.

$$- t = (r)_i \text{ for some } i \in \{0, 1\}$$

Then, if $M \succ^* N$ where $(r)_{\bar{i}} \in N$ with multiplicity at least 2 (for $\bar{i} \in \{0, 1\}, \bar{i} \neq i$), then $M \succ^* P$ where $P = (N \setminus \{(r)_0, (r)_1, (r)_0, (r)_1\}) \cup \{r, r\}$ and by the inductive hypothesis, P is not unitary so M is not unitary.

Otherwise, $(r)_i \in \Phi(M)$ implying that M is not unitary.

$$- t = sr$$

Then, $M \succ^* N$ where $N = (M \setminus \{t, t\}) \cup \{s, r, s, r\}$ and by the inductive hypothesis, N is not unitary, so M is not unitary.

□

Corollary 4.4.33. *Let $M \in \Upsilon$ such that $t \in M$ is a term and $r \in M$ is a term such that t is a proper subterm of r . Then, M is not unitary.*

Proof. If $M \succ^* N$ where $N = (M \setminus \{r\}) \cup \{t\}$ then by the previous lemma, N is not unitary, implying M is not unitary.

Otherwise, t must be a proper subterm of a term s such that $s \in \Phi(M)$ implying that M is not unitary. □

Theorem 4.4.34. *Let $L \in \Upsilon$ be unitary and let x be a variable occurring in L . Let $M \subseteq L$ be the set of terms containing x and let N be a non-empty proper subset of M . Then, $L' = (L \setminus M) \cup N$ is non-unitary.*

Proof. Assume on the contrary that L' is unitary. Then $\Phi(L')$ is a multiset of distinct variables. Also, since \succ never deletes variables, $\Phi(L')$ contains x . Then, since N is non-empty and proper, there exists $t \in M \setminus N$ such that t has x as a subterm. By lemma 4.4.32 and corollary 4.4.33, $(M \setminus N) \cup \Phi(L')$ is non-unitary but $L = (L \setminus M) \cup N \cup (M \setminus N) =$

$L' \cup (M \setminus N) \succ^* \Phi(L') \cup (M \setminus N)$ implying that L is non-unitary which is a contradiction. Therefore, L' is non-unitary. \square

4.4.3 Repeating 5.3.4 and 5.3.6 of Roorda (1991)

Roorda (1991) introduced the notion of “counts for one” along with his description of the PN conditions for well formed proof structures. He then went on to prove several theorems related to his definition of counts for one. This section will seek to reprove these results with the more precise definition of unitary.

Our new definition of $PN(5)$ will be the following:

- **PN(5)** t is unitary.

Theorems 5.3.4 - 5.3.7 from Roorda (1991) are affected by this definition change. Here, we will reprove 5.3.4 and 5.3.6 because they are particularly important and their new proofs are somewhat different than the originals. New proofs of 5.3.5 and 5.3.7 are not difficult.

The following theorem is a proof that any proof structure corresponding to a valid proof in the Lambek calculus satisfies $PN(1-5)$. We will be viewing proof nets using a top down approach that constructs proof nets from axiom links and then inductively from smaller proof nets.

Theorem 4.4.35 (Theorem 5.3.4 in Roorda (1991)). *Every proof net satisfies $PN(1-5)$.*

Proof. The only change is in the definition of **PN(5)** and we will therefore only prove the result for **PN(5)**. We use structural induction on proof net β .

- Base Case: A single axiom link

$$\overline{A \bar{\cdot} x \quad A \dot{\cdot} y}$$

The term assigned to the positive terminal formula is then x which is clearly unitary.

- Inductive Step: We will consider each of the four ways of building a proof net

Vertical ellipsis will represent proof nets ending in the terminal formulae shown below the ellipsis. We will then combine terminal formulae with the given connective to form the new proof net. It is this new proof net which we must prove satisfies $PN(5)$.

- A positive / (or \) step

$$\Gamma \frac{\begin{array}{c} \vdots \\ \beta \bar{\vdash} a \quad \alpha \dagger b \end{array}}{\alpha / \beta \dagger c} \Delta$$

Let t be the term assigned to b through substitutions. Then, we know that the term assigned to c is $\lambda a.t$. Then, by induction, we know that t is unitary. We get that $\lambda a.t$ is unitary by prepending the λ -abstraction reduction on $\lambda a.t$ to the reductions for t .

- A negative \ (or /) step

$$\Gamma \frac{\begin{array}{c} \vdots \\ \alpha \dagger a \quad \beta \bar{\vdash} b \end{array}}{\alpha \backslash \beta \bar{\vdash} d} \gamma \dagger c \quad \Delta$$

Here, Γ and α are the terminal formulae for one proof net and β, γ and Δ are the terminal formulae for the other proof net.

Let s be the term assigned to a through substitutions and let t be the term assigned to c through substitutions. By two applications of the inductive hypothesis, we know that s and t are unitary. In addition, by the way that the variables in the two proof nets are introduced, we know that no variable occurs in both s and t . After the extension, we will replace b by ds throughout t . Since t and s were unitary, this new term will be unitary.

- A negative \bullet step

$$\Gamma \quad \frac{\begin{array}{c} \vdots \\ \alpha \bar{\vdash} a \quad \beta \bar{\vdash} b \end{array}}{\alpha \bullet \beta \bar{\vdash} z} \quad \Delta \quad \gamma \dagger c \quad \Theta$$

Let t be the term assigned to c through substitutions. Then, by induction we know that t is unitary before the introduction of z . Therefore, $t \succ *r$ such that r contains exactly one of a and one of b . Then, when z is introduced, $(z)_0$ is substituted for a and $(z)_1$ is substituted for b . By appending a projection reduction to the sequence of rewrites, we obtain t unitary after the introduction of z .

- A positive \bullet step

$$\Delta \quad \frac{\begin{array}{c} \vdots \\ \beta \dagger a \quad \alpha \dagger b \end{array}}{\alpha \bullet \beta \dagger c} \quad \Gamma$$

Here, Δ and β are the terminal formulae for one proof net and α and Γ are the terminal formulae for the other proof net.

Let s be the term assigned to a through substitutions and let t be the term assigned to b through substitutions. By two applications of the inductive hypothesis, we get that s and t are unitary. Then, we know that $\langle s, t \rangle$ is assigned to c through substitutions. By one application of the pairing rewrite rule, the rewrites on s and t and the fact that s and t cannot share any variables because of the way fresh variables are chosen in proof nets, we get that $\langle s, t \rangle$ is unitary.

□

We need the following definitions from Roorda (1991) before we can begin our reproof of theorem 5.3.6:

Definition 4.4.36. A *bar* b is a list of formula occurrences in a proof frame such that every path from a terminal formula to an axiomatic formula contains exactly one element of b .

Definition 4.4.37. Given a bar b , b_m (b_p) is the sublist of b consisting of exactly the negative (positive) formulae occurrences.

Theorem 4.4.38 (Theorem 5.3.6 in Roorda (1991)). *The condition PN(5) is already fulfilled by terms originating from proof structures satisfying the other conditions.*

Proof. We will step through the necessary modifications.

Lemma 4.4.39 (Lemma 7.3.2 in Roorda (1991)). *For every bar b in a proof frame, b_m and b_p are unitary.*

Proof. First the minus terms. Our proof will use structural induction.

- Base Case:

We know that the bar consisting of all terminal minus formulae is unitary because they are introduced as distinct.

- Inductive Step:

Let b be a bar that is not composed of the terminal formulae. Then, there exists some proof frame building step which builds b out of some b' . By the inductive hypothesis, there exists a sequence of rewrites that rewrite the terms of b'_m to terms that have exactly one occurrence of each variable not contained within a projection.

The following four replacements may occur:

1. $\bar{t} \rightarrow \overset{+}{u} \bar{t}u$

Since u is introduced fresh, the same set of rewrites for b'_m can be used on b_m .

$$2. \lambda u.v \rightarrow \bar{u} \bar{v}$$

The only variable in b_m is u and it was introduced fresh here. Therefore, we can use the same set of rewrites as for b'_m .

$$3. \langle u, v \rangle \rightarrow \bar{u} \bar{v}$$

There are no minus variables, so the same set of rewrites can be used for b_m as for b'_m .

$$4. \bar{t} \rightarrow (\bar{t})_0 (\bar{t})_1$$

To the set of rewrites for b'_m , we will add one of the form $(t)_0, (t)_1 \succ t$. This will show that b'_m is still unitary.

Now, in a similar way, we prove the result for the plus terms.

- Base Case:

We know that the bar consisting of all axiomatic plus formulae is unitary because they are introduced as distinct and there are no links above them.

- Inductive Step:

Let b be a bar that is not composed of the axiomatic formulae. Then, there exists some proof frame building step which builds b' out of b . By the inductive hypothesis, there exists a sequence of rewrites that rewrite the terms of b'_p to terms that have exactly one occurrence of each variable not contained within a projection.

The following four replacements may occur:

$$1. \bar{t} \rightarrow \bar{u} \bar{t} u$$

The same set of rewrites for b'_p can be used for b_p except that any which rewrite subterms of u are removed.

$$2. \lambda u.v \rightarrow \bar{u} \bar{v}$$

The same set of rewrites for b'_p can be used for b_p except that we need one additional rewrite of the form $\lambda u.v \succ v$.

$$3. \langle u, v \rangle^+ \rightarrow \overset{+}{u} \overset{+}{v}$$

The exact same set of rewrites can be used for b_p as can be used for b'_p .

$$4. \bar{t} \rightarrow (\bar{t})_0 (\bar{t})_1$$

There are no plus terms, so we can use the same set of rewrites for b_p as for b'_p .

□

The remainder of the proof remains unchanged and the result is proven.

□

4.5 Derivability conditions

We are now in a position to introduce the correctness criteria for LC graphs with product. They should be considered in contrast to the PN conditions for proof nets and the I conditions for LC graphs without product in section 3.1.3. These correctness criteria will be shown to be sound and complete in 4.6. They are as follows:

- **I(1)** There is a unique node in G with in-degree 0, from which all other nodes are path accessible.
- **I(2)** G is acyclic.
- **I(3)** For every lambda node $v \in V$, there is a path from its plus daughter u to its minus daughter w .
- **I(4)** For every lambda node $v \in V$ with minus daughter w and arbitrary vertex x , either every path from x to w contains v or $v \rightsquigarrow x$.

To extend these correctness criteria to $L^{\bullet*}$, we must add one more condition:

- **I(CT)** For every lambda node $v \in V$, there is a path in G , $v \rightsquigarrow x$, where x is a terminal node and there is no lambda node $v' \in V$ such that $v \rightsquigarrow v' \rightarrow x$.

4.6 I is complete and sound

Throughout this section, we will make the assumption that we are given a sequent and a proof structure with a planar axiomatic linkage. We will be proving the equivalence of the PN conditions from section 3.1.3 and the I conditions from the previous section.

4.6.1 Soundness

Proposition 4.6.1. *If the LC graph for a proof structure satisfies $I(2)$ then the proof structure satisfies $PN(2)$.*

Proof. As in Penn (2001). □

Given this result, we can assume that $PN(2)$ is satisfied for the remainder of our soundness proofs. Thus, variable substitution beginning with the variable labeling the positive terminal formula produces a finite term. Let this term be t .

Proposition 4.6.2. *If the LC graph for a proof structure satisfies $I(3)$ and $I(4)$ then the proof structure satisfies $PN(3)$.*

Proof. Let $\lambda v.s$ be a subterm of t . We know that for this to occur, $[\lambda v.s]$ must be a substitution of the proof frame for some lambda node l as a result of some positive occurrence of either a \backslash or a $/$ rule or as the positive terminal formula. In either case, let v be the variable labeling the negative premise and let u be the variable labeling the positive premise.

Given $I(3)$, we know that there exists a path from u to v . Therefore, when the variable substitution introduces the term $\lambda v.u$, the substitution of u will contain an occurrence of v . Thus, s will contain an occurrence of v .

Then, assume that there exists some occurrence of v outside of s . This means that there exists a path from t to v which does not pass through u and therefore does not pass through l . By $I(4)$, then $l \rightsquigarrow t$ yielding a contradiction.

Therefore, $I(3)$ and $I(4)$ imply $PN(3)$. □

Proposition 4.6.3. *If the LC graph for a proof structure satisfies $I(1)$ then the proof structure satisfies $PN(4)$.*

Proof. By $I(1)$, all nodes are path accessible from the unique proper root node. Since variable substitution begins with the proper root node, and since by proposition 4.3.4, minus nodes have out-degree 0, applying all substitutions until no more can be done will result in a term that contains at least one instance of every minus node. Therefore, every variable assigned to a terminal minus formula occurs in t . □

Proposition 4.6.4. *If a proof structure satisfies $PN(1)$, $PN(2)$, $PN(3)$, $PN(4)$ and $PN(6)$ then it satisfies $PN(5)$.*

Proof. As in Roorda (1991) and reproven as proposition 4.4.38. □

Note that $PN(6)$ follows from the planar construction of the linkage.

4.6.2 Completeness

Proposition 4.6.5. *If a proof net satisfies $PN(3)$, then its LC graph satisfies $I(3)$ and $I(4)$.*

Proof. Given $PN(3)$, suppose the corresponding LC graph has a λ -node l . According to our definition of LC Graphs, such a node corresponds to a subterm $\lambda v.s$ of t . Also, the

minus nodes that are path accessible from l 's minus daughter u represent the occurrences of variables in the term s .

Therefore, since v occurs in s , there must be a path from u to v . Thus, $I(3)$ is satisfied.

Then, let x be an arbitrary node. Assume contrary to $I(4)$, that it is neither the case that every path from x to v contains l nor that $l \rightsquigarrow x$. This implies that v occurs as a subterm of x and that x is not contained within the subterm $\lambda v.s$. \square

Proposition 4.6.6. *If a proof structure satisfies $PN(1)$ and $PN(2)$, then no node in its LC graph that is path accessible from the label of the positive terminal formula is contained in a cycle.*

Proof. If a node v that is path accessible from the label of the positive terminal formula is contained in a cycle, then variable substitution, which begins at that label, would not terminate since it would necessarily contain the term represented by node v , which would yield a subterm that is the same term upon variable substitution. \square

Definition 4.6.7. Given a formula occurrence $\alpha \overset{i}{:} r$ in a proof frame (where i is either positive or negative), the *reflection* of $\alpha \overset{i}{:} r$ is the set of formulae occurrences x such that each x is a descendant of $\alpha \overset{i}{:} r$ and on the path from $\alpha \overset{i}{:} r$ to x , all formulae have polarity i (including x). In other words, the reflection of $\alpha \overset{i}{:} r$ is the set of its i path descendants.

Definition 4.6.8. A formula $\alpha \overset{i}{:} r$ in a proof frame is *peripheral* to a reflection if the parent of $\alpha \overset{i}{:} r$ is in that reflection but $\alpha \overset{i}{:} r$ is not.

Example 4.6.9. In figure 4.3, the formulae whose annotations are c , d , e and g are all in the reflection of $((A/(A \bullet A)) \bullet A) \overset{+}{:} c$. Then, $(A \bullet A) \overset{-}{:} f$ is peripheral to that reflection. Similarly, $A \overset{-}{:} ab$ and $(A/A) \overset{-}{:} a$ are in the reflection of $(A/A) \overset{-}{:} a$ and $A \overset{+}{:} b$ is peripheral to it.

Proposition 4.6.10. *If a proof structure satisfies $PN(1)$, $PN(2)$, $PN(3)$ and $PN(4)$ then its LC graph satisfies $I(1)$.*

Figure 4.3: The proof frame for $(A/A) \vdash (A/(A \bullet A)) \bullet A$ (repeat of figure 4.1).

$$\begin{array}{c}
\frac{A \bar{\vdash} (f)_0 \quad A \bar{\vdash} (f)_1}{(A \bullet A) \bar{\vdash} f} \quad A \dagger \vdash g \\
\frac{\frac{A \bar{\vdash} ab \quad A \dagger \vdash b}{(A/A) \bar{\vdash} a} \quad \frac{A \dagger \vdash d}{((A/(A \bullet A)) \bullet A) \dagger \vdash c}}{(A/(A \bullet A)) \dagger \vdash e} \\
\text{Substitutions} := [c := \langle e, d \rangle], [e := \lambda f.g]
\end{array}$$

Proof. To establish the validity of $I(1)$, we first show that there is a unique node with in-degree 0. By $PN(1)$, there is a unique positively labeled terminal formula. By proposition 4.3.3, no other node has in-degree 0. If every node is accessible from this one, then by $PN(2)$, it must have in-degree 0. So it remains only to show that every node is accessible from this.

To prove accessibility, we will consider vertices in our LC Graph (which are equivalent to variables in the proof frame) in two categories:

1. Variables labeling formulae in the reflection of the positive terminal formula

By the definition of LC Graphs, we know that from any positive formula with children, there are edges in the graph from the label of the parent to the labels of the children. Thus, since the positive terminal formula is the ancestor of all the formulae in its reflection, all variables labeling those formulae are accessible from the root node.

All other variables must have been introduced with a negative formula below them.

2. Variables labeling formulae with a negative formula ancestor in the proof frame

Let a be such a variable. We will proceed to prove the result using structural induction on the proof frame. However, we will need a stronger inductive hypothesis.

Inductive Hypothesis: Every such node a is path accessible from the node corresponding to the positively labeled terminal formula and if a is a minus node, it is

accessible via a path ending in an edge introduced due to an axiom link (and not a positive rule substitution).

- Base Case:

The base cases consist of labels of negative terminal formulae and negative formulae that are peripheral to the reflection of the positive terminal formula.

Let m be a label of a negative terminal formula. By $PN(4)$, we know that m occurs in t . In addition, since m is not a minus daughter of a lambda node the only possible in-edges are generated by axiom links.

Let m be a label of a negative formula which is peripheral to the reflection of the positive terminal formula. Let p be its positive sibling and let l be its parent. We know that m must be the minus daughter of a lambda node. By $PN(4)$ and proposition 4.6.5, we know that $I(3)$ holds which means that there exists a path from p to m . Also, if l is on this path, then a cycle exists from $l \rightarrow p \rightsquigarrow l$. We already know from the first case shown above that l is path accessible from the root node and by proposition 4.6.6 this is impossible. Therefore, the path from p to m must be through another in-edge of m , but the only other in-edges are generated by axiom links.

- Inductive Step:

Consider any other variable. It must have been introduced by a rule. By the inductive hypothesis, any variables occurring as labels of the conclusion of that rule are accessible by paths from the root and if they are negative a path exists whose last edge is due to an axiom link.

We will consider the rules in four cases:

(a) The last step is one of

$$- \frac{\beta \dagger v' \quad \alpha \bar{\vdash} u}{\alpha \setminus \beta \dagger v}$$

$$- \frac{\beta \bar{\cdot} u \quad \alpha \dagger \cdot v'}{\alpha/\beta \dagger \cdot v}$$

By the inductive hypothesis, there is a path from the root node to v . By the definition of LC Graphs, there is a path from v to v' . Then, by $I(3)$, there is a path from the v' to u . The last step in this path must be through an axiom link for the same reason as in the second base case.

(b) The last step is

$$- \frac{\beta \dagger \cdot v' \quad \alpha \dagger \cdot v''}{\alpha \bullet \beta \dagger \cdot v}$$

By the inductive hypothesis, v is accessible and by the definition of LC Graphs, both children are accessible.

(c) The last step is one of

$$\begin{aligned} & - \frac{\alpha \dagger \cdot u \quad \beta \bar{\cdot} su}{\alpha \setminus \beta \bar{\cdot} s} \\ & - \frac{\alpha \bar{\cdot} su \quad \beta \dagger \cdot u}{\alpha/\beta \bar{\cdot} s} \end{aligned}$$

By proposition 4.3.1 we know that s contains a unique minus variable m and by the inductive hypothesis, there is a path from the root node to m with a last step that is due to an axiom link. Let $\alpha \bar{\cdot} m$ be the unique formula occurrence in the proof frame such that m is the label. Then, let M be the set of axiomatic formula occurrences that are negative path descendants of $\alpha \bar{\cdot} m$ and let \bar{M} be the set of negative axiomatic formulae. Let P be the set of positive axiomatic formulae with axiom links to M . We now want to prove that each of the variables labeling formulae in P are path accessible from the root.

We know that m is accessible by a path from the root ending in an axiomatic linkage which implies that m occurs in t . Then, by $PN(5)$, t is unitary. By lemma 4.4.39 we know that \bar{M} is unitary and by theorem 4.4.34 removing any non-empty proper subset of M from \bar{M} will yield a

non-unitary multiset. Thus, it must be the case that each of the labels in M occurs in t since otherwise, t will be non-unitary. The only way for this to occur is if all of the labels in P are path accessible from the root. Then, since each formula occurrence in M is linked to from a formula in P , each variable in each label in M is path accessible from the root. Finally, the term su must appear in M since it appears in the labels of all negative path descendants of $\gamma \bar{\cdot} su$ where $\gamma = \alpha \setminus \beta$ or $\gamma = \beta / \alpha$. Therefore, u is path accessible from the root node.

(d) The last step is

$$- \frac{\alpha \bar{\cdot} (s)_0 \quad \beta \bar{\cdot} (s)_1}{\alpha \bullet \beta \bar{\cdot} s}$$

No new variables are introduced and therefore the theorem is vacuously true.

□

Proposition 4.6.11. *If the LC graph for a proof structure satisfies I(1) and PN(2) then its proof structure satisfies I(2).*

Proof. By proposition 4.6.6, no node which is path accessible from the label of the positive terminal formula is contained in a cycle and by I(1), every node is so accessible. □

4.6.3 Equivalence of PN(CT) and I(CT)

Proposition 4.6.12. *If the LC graph for a proof structure satisfies I(2), I(3) and I(4) then it satisfies I(CT) if and only if its proof structure satisfies PN(CT).*

Proof. (\Rightarrow)

By $I(CT)$, we know that given a lambda node v with minus daughter w , there is a path $v \rightsquigarrow x$ where x is some terminal node for which there is no lambda node $v' \in V$ such that $v \rightsquigarrow v' \rightarrow x$. By proposition 4.3.5, we know that x is a minus node.

If x is not the minus daughter of a lambda node, then x corresponds to the label of a negative terminal formula and therefore, v trivially does not expand to a closed term.

Otherwise, suppose that every such x is the minus daughter of a lambda node. That is, every such x corresponds to the bound variable of some lambda term. Let v' be the lambda node such that x is its minus daughter.

By $I(4)$, either $v \rightsquigarrow v'$ or $v' \rightsquigarrow v$. If $v \rightsquigarrow v'$, then $I(CT)$ is violated. Therefore, $v' \rightsquigarrow v$. This means that there is a free occurrence of x in v , satisfying $PN(CT)$.

(\Leftarrow)

Given $PN(CT)$, no lambda term l is a closed term. Therefore, from any lambda node v , there must be a path to a minus node other than the minus daughters of the lambda nodes that are reachable from v . Thus, $I(CT)$ holds.

□

Chapter 5

LC Graphs and the NP

Completeness of L^\bullet

LC Graphs have given us some insight into what the precise difference is between L^\bullet and L . We would like to use these insights to help us determine the computational complexity of sequent derivability in L . One possibility is that L is *NP*-complete and a proof similar to that of Pentus (2003) for L^\bullet can be used to prove this. Penn (2001) has remarked on the unlikeliness of this based on the fact that no equivalent sequents were forthcoming in his search.

This section will proceed by partially reproving the result of Pentus (2003). We will use *LC*-graphs to delve more deeply into the possibility of altering that proof to apply to L . A by-product of this illustrates the importance of $I(4)$.

5.1 The reduction

Much of this section is taken directly from Pentus (2003) although the notion of “coats” is taken from Penn (2001).

Let $c_1 \wedge \dots \wedge c_m$ be a boolean formula in conjunctive normal form with clauses c_1, \dots, c_m and variables x_1, \dots, x_n . We now define a set of formulae which will be used to build the

Table 5.1: The two coats of the reduction.

True Coat	$(A \setminus F) \bullet B$
False Coat	$A \setminus (F \bullet B)$

desired sequent.

Each formula in the set will consist of a simple formula which is then wrapped in m “coats”, one for each clause. The two possible coats that can be wrapped around a formula F can be seen in table 5.1. In that table, F is an arbitrary formula and A and B are Lambek atoms.

Then, in this spirit, we define three series of formulae. Let $1 \leq i \leq n$, let $1 \leq j \leq m$ and let $t \in \{0, 1\}$. In other words, let x_i be a variable, let c_j be a clause and let t be a truth assignment for x_i . We will follow Pentus (2003) in using \neg_0 to represent \neg and using \neg_1 to represent a lack of negation. That is, $\neg_0 x_i = \neg x_i$ and $\neg_1 x_i = x_i$.

$$\begin{aligned}
E_i^0(t) &= p_{i-1}^0 \setminus p_i^0 \text{ for } t \in \{0, 1\} \\
E_i^j(t) &= (p_{i-1}^j \setminus E_i^{j-1}(t)) \bullet p_i^j \text{ if } \neg_t x_i \text{ appears in } c_j \\
E_i^j(t) &= p_{i-1}^j \setminus (E_i^{j-1}(t) \bullet p_i^j) \text{ otherwise} \\
G^0 &= p_0^0 \setminus p_n^0 \\
G^j &= (p_0^j \setminus G^{j-1}) \bullet p_n^j \\
H_i^0 &= p_{i-1}^0 \setminus p_i^0 \\
H_i^j &= p_{i-1}^j \setminus (H_i^{j-1} \bullet p_i^j)
\end{aligned}$$

To summarize the above formulae, each of the E , G and H series begin with a simple formula. The G series then wraps that formula in true coats, the H series wraps that formula in false coats and the E series wraps that formula with a coat depending on

which variables occur in which clause.

We are only interested in the formulae in this set which consist of m coats so we introduce the following shorthands:

$$\begin{aligned} E_i(t) &= E_i^m(t) \\ G &= G^m \\ H_i &= H_i^m \\ F_i &= (E_i(1)/H_i) \bullet H_i \bullet (H_i \setminus E_i(0)) \end{aligned}$$

Then, the sequent that is assigned to $c_1 \wedge \dots \wedge c_m$ by the reduction is $F_1, \dots, F_n \vdash G$.

5.2 Correctness of the reduction

The following theorem appears in another form in Pentus (2003) and we will not go into the details of its proof.

Theorem 5.2.1. *$F_1, \dots, F_n \vdash G$ is derivable in L^\bullet (and $L^{*\bullet}$) if and only if $E_1(t_1), \dots, E_n(t_n) \vdash G$ is derivable in \bullet (and $L^{*\bullet}$) for some $\langle t_1, \dots, t_n \rangle \in \{0, 1\}^n$.*

Proof. The “only if” direction is proven in lemma 3.16 in Pentus (2003).

The “if” direction is a direct consequence of lemma 3.3 in Pentus (2003) and n uses of the *CUT* rule. □

Intuitively, this theorem says that of the two side H ’s competing for the middle H in F , one must get it and the other must cancel with $E_i(t)$ for $t \in \{0, 1\}$, leaving the other $E_i(t')$. The consequence of this theorem is that derivability of $F_1, \dots, F_n \vdash G$ is dependent only on sequents of the form $E_1(t_1), \dots, E_n(t_n) \vdash G$.

5.2.1 Proof nets of $E_1(t_1), \dots, E_n(t_n) \vdash G$

This section will discuss the proof nets of the sequent $E_1(t_1), \dots, E_n(t_n) \vdash G$. Examples of the proof nets of two such sequents are given in figures 5.11 and 5.12.

Because the sequent $E_1(t_1), \dots, E_n(t_n) \vdash G$ is parameterized, we can only consider its proof frame in pieces. Because the recursive definitions of the E and G series have 5 parts, the proof frame is built using 5 schemata found in figures 5.1 - 5.5.

Figure 5.1: The proof frame for $E_i^0(t) = p_{i-1}^0 \setminus p_i^0$.

$$\frac{p_{i-1}^0 \dagger a_i^0 \quad p_i^0 \bar{\vdash} \alpha a_i^0}{p_{i-1}^0 \setminus p_i^0 \bar{\vdash} \alpha}$$

Figure 5.2: The proof frame for $(p_{i-1}^j \setminus E_i^{j-1}(t)) \bullet p_i^j \bar{\vdash}$.

$$\frac{\frac{p_{i-1}^j \dagger a_i^j \quad E_i^{j-1}(t) \bar{\vdash} (\alpha)_0 a_i^j}{p_{i-1}^j \setminus E_i^{j-1}(t) \bar{\vdash} (\alpha)_0} \quad p_i^j \bar{\vdash} (\alpha)_1}{(p_{i-1}^j \setminus E_i^{j-1}(t)) \bullet p_i^j \bar{\vdash} \alpha}$$

Figure 5.3: The proof frame for $p_{i-1}^j \setminus (E_i^{j-1}(t) \bullet p_i^j) \bar{\vdash}$.

$$\frac{p_{i-1}^j \dagger a_i^j \quad \frac{E_i^{j-1}(t) \bar{\vdash} (\alpha)_0 a_i^j \quad p_i^j \bar{\vdash} (\alpha)_1 a_i^j}{p_{i-1}^j \setminus E_i^{j-1}(t) \bar{\vdash} \alpha a_i^j}}{p_{i-1}^j \setminus (E_i^{j-1}(t) \bullet p_i^j) \bar{\vdash} \alpha}$$

In addition to these pieces, we will also label the (negative) terminal formula $E_i(t_i)$ with u_i and the (positive) terminal formula G with u . This completely specifies the lambda variables annotated to the proof frame.

Figure 5.4: The proof frame for $G^0 = p_0^0 \setminus p_n^0$.

$$\frac{p_n^0 \dagger b^0 \quad p_0^0 \bar{\vdash} d^0}{p_0^0 \setminus p_n^0 \dagger \alpha}$$

$$\text{Substitutions} = [\alpha := \lambda d^0 . b^0]$$

Figure 5.5: The proof frame for $G^j = (p_0^j \setminus G^{j-1}) \bullet p_n^{j-}$.

$$\frac{p_n^j \dagger b^j \quad \frac{G^{j-1} \dagger e^j \quad p_0^j \bar{\vdash} d^j}{p_0^j \setminus G^{j-1} \dagger c^j}}{(p_0^j \setminus G^{j-1}) \bullet p_n^j \dagger \alpha}$$

$$\text{Substitutions} = [\alpha := \langle c^j, b^j \rangle], [c^j := \lambda d^j . e^j]$$

Claim 5.2.2. *Let $0 \leq i \leq n$ and $0 \leq j \leq m$ so that p_i^j is a Lambek atom in $E_1(t_1), \dots, E_n(t_n) \vdash G$. Then in the proof frame for $E_1(t_1), \dots, E_n(t_n) \vdash G$ there is exactly one positive and one negative axiomatic formula p_i^j .*

Proof. We prove the claim using three cases:

- $1 \leq i \leq n - 1$ and $0 \leq j \leq m$

It is clear that p_i^j does not appear in G because only p_i^j such that $i = 0$ or $i = n$ appear in G . In addition, p_i^j appears exactly once in $E_i(t_i)$ and exactly once in $E_{i+1}(t_{i+1})$ and no where else. The occurrence in $E_i(t_i)$ occurs to the left of the \setminus and according to figures 5.2 and 5.3, this results in a positive axiomatic formula. Similarly, the occurrence in $E_{i+1}(t_{i+1})$ occurs to the right of the \bullet and according to figures 5.2 and 5.3, this results in a negative axiomatic formula.

- $i = 0$ and $0 \leq j \leq m$

Then, there is exactly one occurrence of p_i^j in each of $E_0(t_0)$ and G and no others.

Also, according to figures 5.2 and 5.3 the occurrence in $E_0(t_0)$ results in a positive axiomatic formula and according to figure 5.5, the occurrence in G results in a negative axiomatic formula.

- $i = n$ and $0 \leq j \leq m$

Then, there is exactly one occurrence of p_i^j in each of $E_n(t_n)$ and G and no others.

Then, according to figures 5.2 and 5.3 the occurrence in $E_0(t_0)$ results in a negative axiomatic formula and according to figure 5.5, the occurrence in G results in a positive axiomatic formula.

□

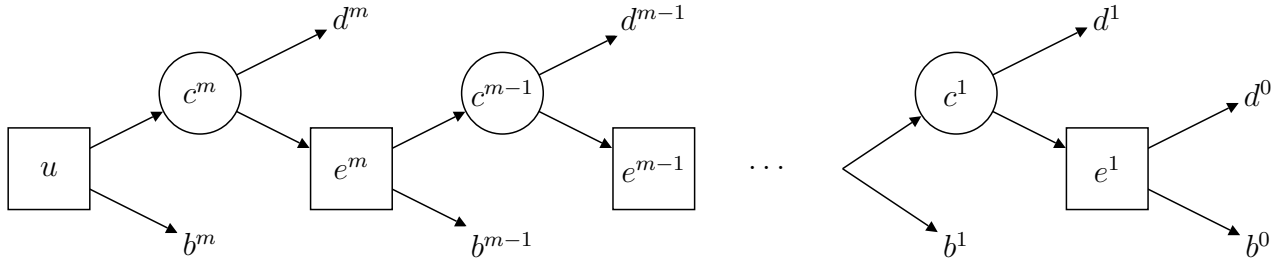
Because the p_i^j each have a single unique positive and negative occurrence as axiomatic formulae, we will refer to these as the positive axiomatic formula p_i^j and negative axiomatic formula \bar{p}_i^j .

Claim 5.2.3. *Let l be the lambda term labeling the negative axiomatic formula \bar{p}_i^j for $1 \leq i \leq n$ and $1 \leq j \leq m$. Then, l necessarily contains the lambda variables $u_i, a_i^{j+1}, \dots, a_i^m$. In addition, l contains a_i^j if and only if $\neg_{t_i}x$ appears in c_j .*

Proof. The only rule schemata that are used below the negative axiomatic formula \bar{p}_i^j are those in figures 5.2 and 5.3. The schemata indicate that any variable which annotates the conclusion of the schema must also annotate all negative premises of the schema. Therefore, since u_i annotates $E_i(t_i)$ in the proof frame, u_i annotates \bar{p}_i^j in the proof frame. Furthermore, a_k^l appears in the term annotating $E_k^{l-1}(t)$ for all $1 \leq k \leq n$ and $1 \leq j \leq m$ and therefore, annotates all \bar{p}_i^r such that $1 \leq r \leq k$. Finally, it is clear from figures 5.2 and 5.3 and the definition of the E series for formulae that the annotation of \bar{p}_i^j contains a_i^j if and only if $\neg_{t_i}x$ appears in c_j . □

Claim 5.2.4. *Let l be the lambda term labeling the negative axiomatic formula \bar{p}_i^0 for $1 \leq i \leq n$. Then, l necessarily contains the lambda variables u_i, a_i^0, \dots, a_i^m .*

Figure 5.6: LC graph of substitutions of $E_1(t_1), \dots, E_n(t_n) \vdash G$.



Proof. This proof is similar to the one above. □

Then, due to claim 5.2.2 there is exactly one linkage over the proof frame for $E_1(t_1), \dots, E_n(t_n) \vdash G$.

Claim 5.2.5. *The unique linkage over the proof frame for $E_1(t_1), \dots, E_n(t_n) \vdash G$ is always planar.*

Proof. This follows from a simple (but tedious) inductive proof in the same spirit as the above proofs. □

5.2.2 The LC graph for $E_1(t_1), \dots, E_n(t_n) \vdash G$

We can now begin to analyze the unique LC graph of $E_1(t_1), \dots, E_n(t_n) \vdash G$ piece by piece with a total of 4 different pieces.

The substitutions

Due to the substitutions from figures 5.4 and 5.5, we get the partial LC graph in figure 5.6.

The out-neighbourhood of b_j

The next piece of the LC graph we will examine is the out-neighbourhood of b_j for $0 \leq j \leq m$. We can see from the rule schemata in figures 5.4 and 5.5 that the b^j label the

Figure 5.7: Out-neighbourhoods of b^j for $0 \leq j \leq m$.

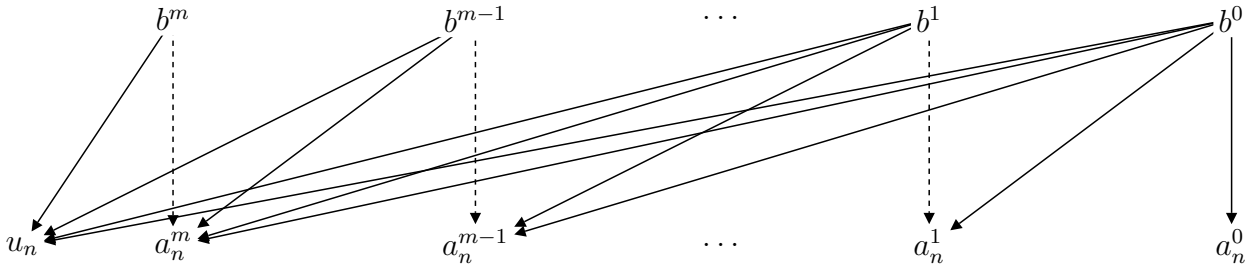
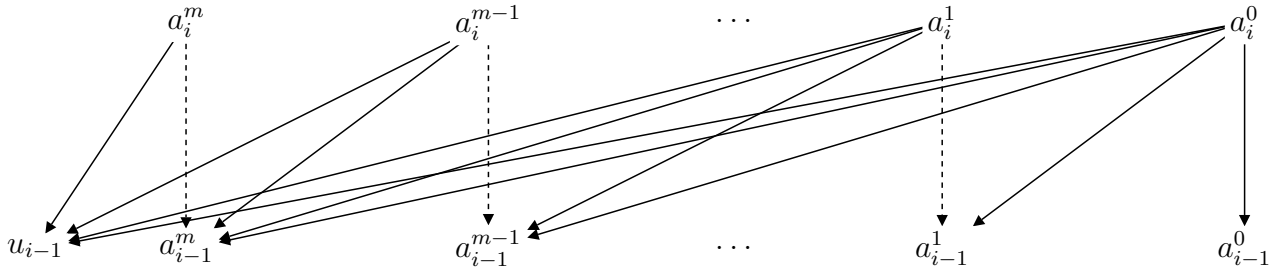


Figure 5.8: Out-neighbourhoods of a_i^j for $0 \leq j \leq m$.



positive axiomatic formula p_n^j . We know that the negative axiomatic formulae p_n^j occur in $E_n(t_n)$. Then, by claim 5.2.3 we know that the out neighbourhood of b_j is precisely $u_n, a_n^{j+1}, \dots, a_n^m$ plus a_n^j if and only if $\neg_{t_n} x_n$ appears in c_j . We get the partial LC graph in figure 5.7 with dotted lines indicating the edges which may or may not occur depending on the appearance of $\neg_t x_i$ in c_j .

The out-neighbourhood of a_i^j

Now, we consider the out-neighbourhoods of a_i^0 through a_i^m for $1 < i \leq n$. Like before, a_i^j labels the positive axiomatic formula p_{i-1}^j . According to claim 5.2.3, we get the partial graph in figure 5.8.

The out-neighbourhood of a_i^0

Finally, we wish to determine the out-neighbourhoods of a_1^0, \dots, a_m^0 . It is easy to see from figures 5.2 and 5.3 that a_1^j labels the positive axiomatic formula p_0^j . Also, it is clear from figure 5.5 that the negative axiomatic formula with p_0^j has the lambda variable d^j as its label.

The big picture

Putting all four of these sections together, we get the graph in figure 5.9.

The integrity conditions

We are now in a position to evaluate the derivability of $E_1(t_1), \dots, E_n(t_n) \vdash G$ based on its LC-graph and the integrity conditions introduced in chapter 4.

- $I(1)$ and $I(2)$

It is easy to check that no matter which of the dashed edges are present, this graph is acyclic and that all nodes are path accessible from t .

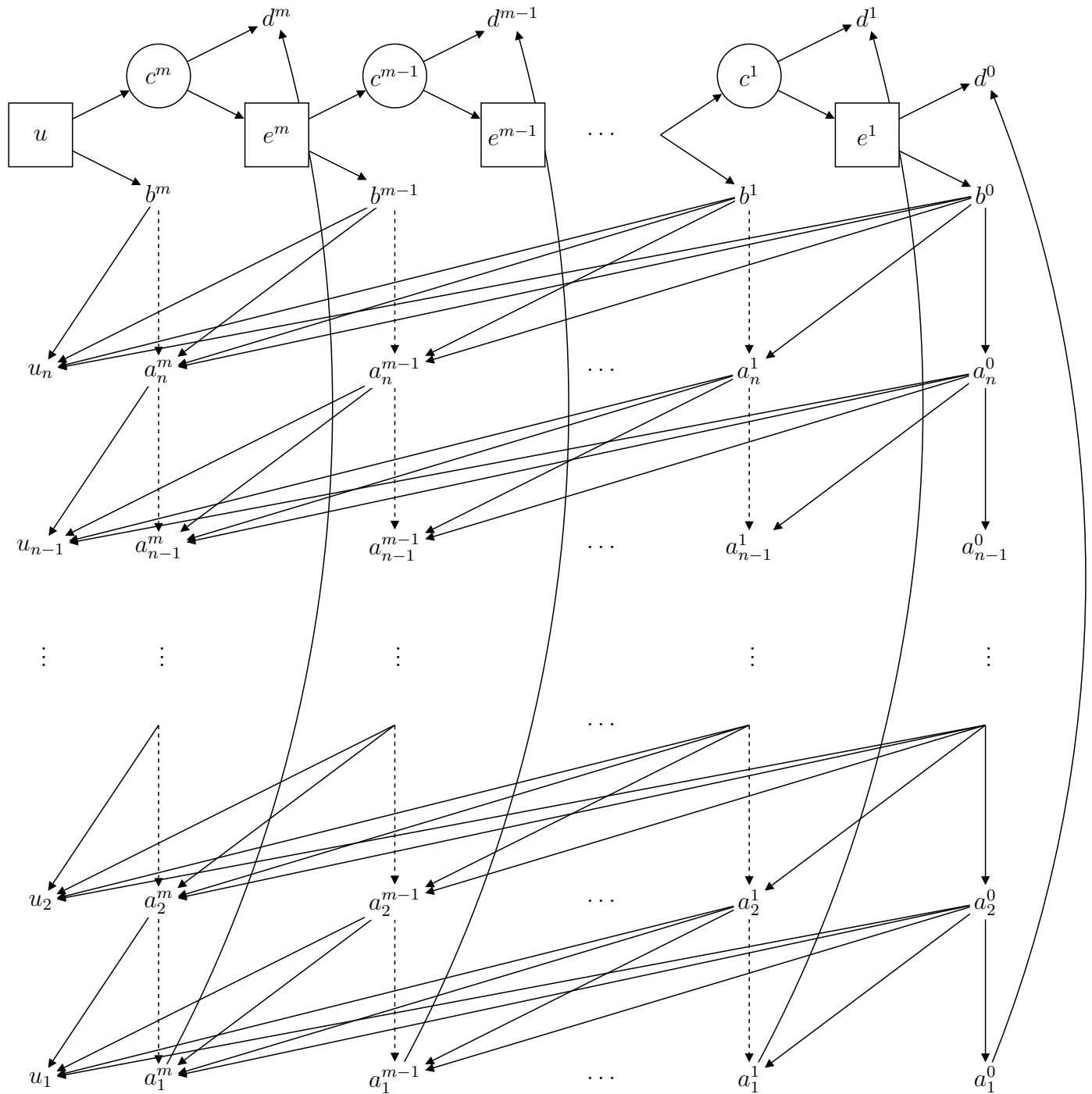
- $I(3)$

To determine whether this LC graph satisfies $I(3)$, we must check whether there is a path from e^j to d^j for $1 \leq j \leq m$ and from b^0 to d^0 . Clearly b^0 is path accessible from any e^j and furthermore, a_2^0 is path accessible from b^0 . Then, a_1^j is accessible from a_2^0 and therefore, $I(3)$ is always satisfied.

- $I(4)$

It is not difficult to see that the only way that $I(4)$ can be violated is if there is a path from $b_j \rightsquigarrow d_j$ for $1 \leq j \leq m$. Furthermore, d_j can be path accessible from b_j if and only if all of the dotted edges in the j column are present. In other words, if

Figure 5.9: LC graph of $E_1(t_1), \dots, E_n(t_n) \vdash G$.



and only if none of the literals $\neg_{t_i}x_i$ for $1 \leq i \leq n$ appear in c_j . This occurs if and only if this truth assignment falsifies $c_1 \wedge \dots \wedge c_m$.

Based on these observations, the only important paths are those from b^j to a_1^j for $1 \leq j \leq m$. A simplified view of figure 5.9 is given in figure 5.10.

Furthermore, we can see that an $I(4)$ violation occurs if and only if there is some j such that $\neg_{t_i}x_i$ does not occur in c_j for any $1 \leq i \leq n$. But this occurs if and only if $c_1 \wedge \dots \wedge c_m$ is unsatisfiable. Thus, the reduction is correct.

Conclusion

It is clear from the correctness proof that the only relevant I condition for these sequents is $I(4)$. Given that in L , $I(4)$ is not required for a proof structure to qualify as a proof net, at the very least it will require radical change to manipulate this proof into one for the NP -completeness of L and at worst, this task will be impossible.

Figures 5.11 and 5.12 are examples of the proof frames for two variable assignments for $x_1 \vee \neg x_2$. Figures 5.13 and 5.14 give the corresponding LC graphs.

Figure 5.10: Simplified view of the LC graph of $E_1(t_1), \dots, E_n(t_n) \vdash G$.

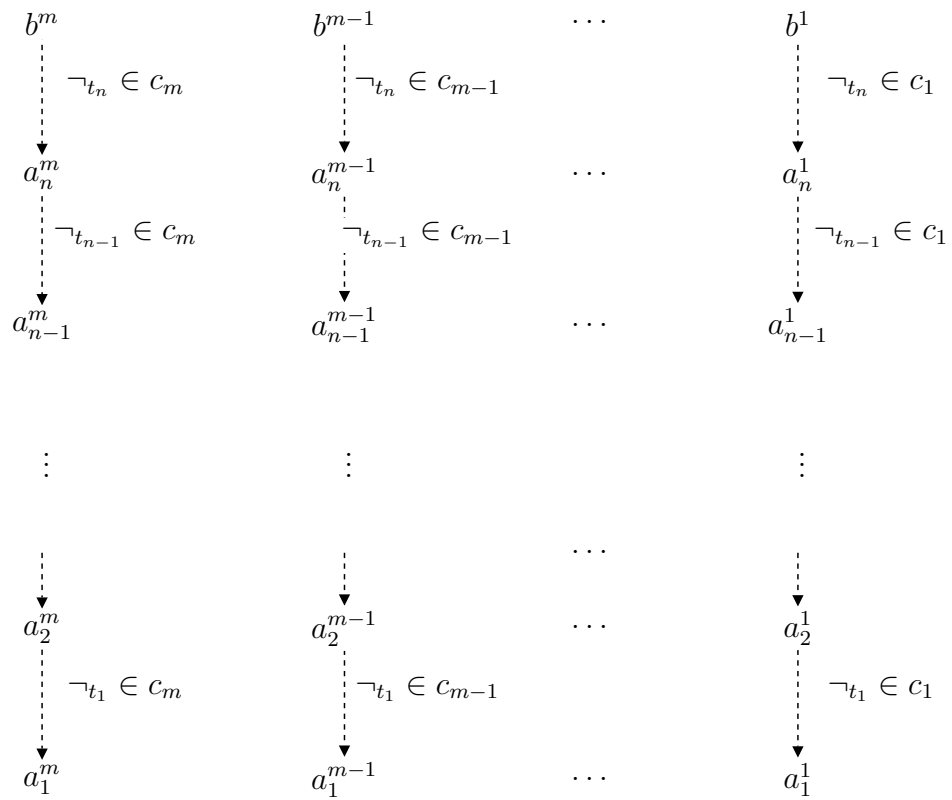
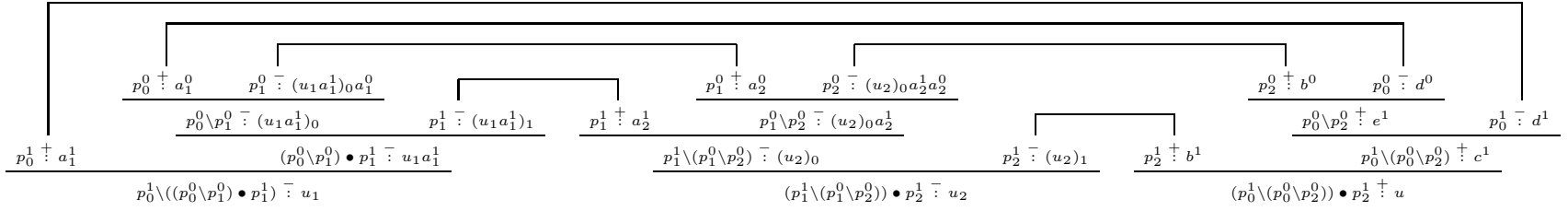
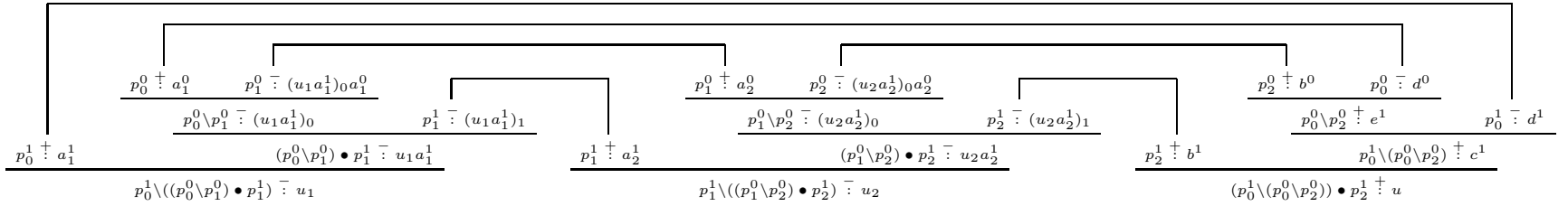


Figure 5.11: Proof structure of $E_1(0), E_2(0) \vdash G$ for $x_1 \vee \neg x_2$.


$$\text{Substitutions} = [u := \langle c^1, b^1 \rangle], [c^1 := \lambda d^1. e^1], [e^1 := \lambda d^0. b^0]$$

 Figure 5.12: Proof structure of $E_1(0), E_2(1) \vdash G$ for $x_1 \vee \neg x_2$.


$$\text{Substitutions} = [u := \langle c^1, b^1 \rangle], [c^1 := \lambda d^1. e^1], [e^1 := \lambda d^0. b^0]$$

Figure 5.13: LC graph of $E_1(0), E_2(0) \vdash G$ for $x_1 \vee \neg x_2$ (see in figure 5.11).

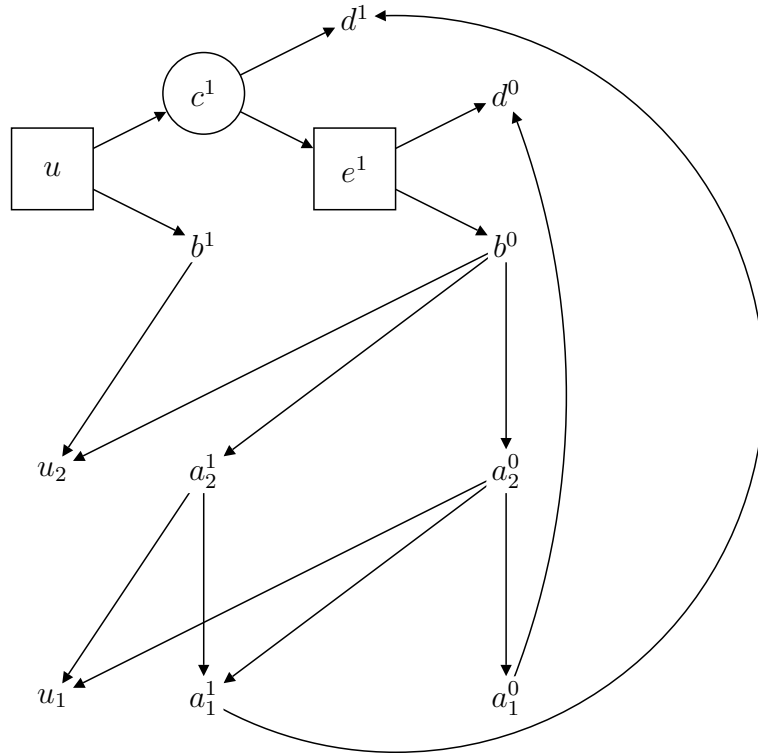
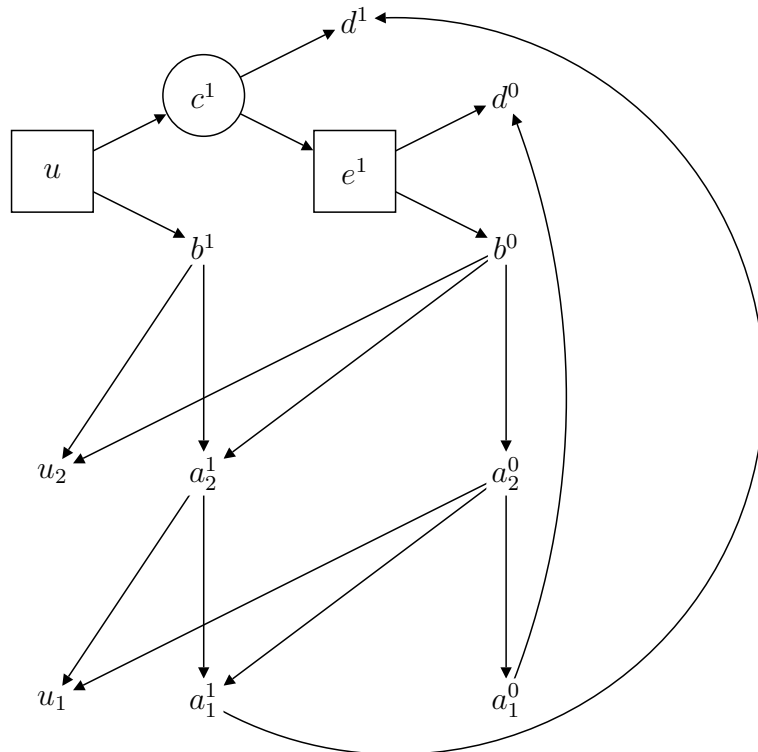


Figure 5.14: LC graph of $E_1(0), E_2(1) \vdash G$ for $x_1 \vee \neg x_2$ (shown in figure 5.12).



Chapter 6

Conclusion

In the introduction we outlined four objectives for this thesis. The first was a simple overview of the existing graph formalisms for representing proof nets.

Second, we sought to find the precise difference between parsing in L^\bullet and L by way of LC graphs. This was done by finding a graph formalism such that the only difference is a single path condition on the LC graph of a proof structure. While this finding does not directly lead us to a polynomial time algorithm it indicates where the difficulties lie in parsing in L^\bullet and shows us what short cuts we can take when parsing in L .

Next, we sought to investigate the NP-completeness of L^\bullet with an eye towards altering the proof of Pentus (2003) to get a proof of NP-completeness for L . We found that in reproving the result, that the NP-completeness shown in the proof is dependent only on the $I(4)$ condition indicating that an alteration to the proof will have to be radical at the very least.

Finally, we sought to unify the varied research results in the field into a cohesive body. This document laid out a standard set of terminology and notation entirely taken from previous work to try to ease understanding of the material.

6.1 Future directions

A polynomial time algorithm for L

An obvious next step in the search for tractability is to find a polynomial time algorithm for L if one exists. Penn (2001) provides some useful results to this end. A chart parsing method similar to the methods used for parsing context free grammars appears to be promising.

Polynomial time algorithms for other simplifications of L^\bullet

Regardless of whether there exists a polynomial time algorithm for L , it is possible that other simplifications of L^\bullet will yield tractable parsing problems. For each posited simplification, parsing algorithms would need to be investigated as well as more practical investigations of whether the calculus is linguistically significant.

Phase transition

Despite being NP-complete, SAT solvers are a significant research area in computer science. This is partly due to the fact that there exist a variety of subsets of the general SAT problem that are particularly difficult to solve and another variety which are particularly easy to solve. For the latter, much research has been done on efficient algorithms for solving such SAT instances.

With this in mind, the question of how the reduction of chapter 5 effects the varying difficulty of the subsets of SAT becomes relevant. A practical investigation as to whether the sequents reduced from difficult SAT instances ever appear in natural language could give hope that parsing in L^\bullet is in fact tractable in all natural language applications despite the general problem being NP-complete.

Bibliography

Eric Aarts. Proving Theorems of the Second Order Lambek Calculus in Polynomial Time. *Studia Logica*, 53:373–387, 1994.

Eric Aarts and Kees Trautwein. Non-associative Lambek Categorical Grammar in Polynomial Time. *Mathematical Logic Quarterly*, 41, 1994.

H. P. Barendregt. *The Lambda Calculus: Its syntax and semantics*. North-Holland, 1981.

Bob Carpenter. *Type-Logical Semantics*. The MIT Press, 1997.

Bob Carpenter and Glyn Morrill. Switch graphs for parsing type logical grammars. *Proceedings of the International Workshop on Parsing Technology*, 2005.

Vincent Danos and Laurent Regnier. The Structure of Multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

Philippe de Groote. An algebraic correctness criterion for intuitionistic multiplicative proof-nets. *Theoretical Computer Science*, 224(1–2):115–134, 1999a.

Philippe de Groote. The Non-associative Lambek Calculus with Product in Polynomial Time. In Neil V. Murray, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes In Computer Science*, pages 128–139. Springer-Verlag, London, UK, 1999b.

- Mario Fadda and Glyn Morrill. The Lambek Calculus with Brackets. In R.A.G. Seely C. Casadio, P.J. Scott, editor, *Language & Grammar: Studies in Mathematical Linguistics and Natural Language*. CSLI Press, Stanford, 2005.
- Jean-Yves Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.
- Francois Lamarche. Proof Nets for Intuitionistic Logic I: Essential Nets. Unpublished manuscript, 1995.
- Francois Lamarche and Christian Retore. Proof nets for the Lambek calculus – an overview. In V. Michele Abrusci and Claudio Casadio, editors, *Proceedings of the 1996 Roma Workshop, Proofs and Linguistic Categories*, pages 241–262. CLUEB, Bologna, 1996.
- Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.
- Richard Moot. Graph Algorithms for Improving Type-Logical Proof Search. *Proceedings of Categorical Grammar 2004*, 2004.
- G. Morrill. Grammar and Logical Types. *Theoria*, LXII(3):260–293, 1990.
- G. Morrill and A. Gavarro. Catalan Clitics. In A. Lecomte, editor, *Word Order in Categorical Grammar*, pages 211–232. Clermont-Ferrand, 1992.
- Gerald Penn. A Graph-Theoretic Approach to Sequent Derivability in the Lambek Calculus. *Language and Computation*, 1(3):1–26, 2001.
- M. Pentus. Product-free Lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660, 1997.
- M. Pentus. Free monoid completeness of the Lambek calculus allowing empty premises. In J. M. Larrazabal, Lascar D., and Mints G., editors, *Logic Colloquium '96: proceedings*

ings of the colloquium held in San Sebastian, Spain, July 9–15, 1996, pages 171–209. Springer, 1998. Lecture Notes in Logic, 12.

M. Pentus. Lambek calculus is NP-complete. CUNY Ph.D. Program in Computer Science Technical Report TR-2003005, CUNY Graduate Center, New York, May 2003. <http://www.cs.gc.cuny.edu/tr/techreport.php?id=79>.

Christian Retore. Perfect matchings and series-parallel graphs: multiplicatives proof nets as R&B-graphs. In M. Okada & A. Scedrov J.-Y. Girard, editor, *Linear '96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, <http://www.elsevier.nl/>, 1996.

Christian Retore. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. *Rapport de recherche 3652*, 1999.

Dirk Roorda. *Resource Logics: Proof-theoretical Investigations*. PhD thesis, Universiteit van Amsterdam, 1991.

Mark Steedman. *The Syntactic Process*. The MIT Press, 2000.

Mary McGee Wood. *A Categorical Syntax for Coordinate Constructions*. PhD thesis, University of London, 1988.