Linear Logic for Linguists

Introductory Course, ESSLLI-00

Dick Crouch
Xerox PARC
crouch@parc.xerox.com

Josef van Genabith
Dublin City University
josef@compapp.dcu.ie

Draft 20 June 2000[1]

[1] This is a rough and very incomplete draft. Bibliographic references within the text are almost entirely missing, and only a preliminary list of further reading is given at the end. It is hoped that these notes will grow into a more polished tutorial on 'linear logic for dummies': comments, suggestions and errata are welcome. A more recent version of these notes may be available at `www.parc.xerox.com/istl/members/crouch`

Contents

Chapter 1

Introduction

1.1 Our Intended Audience

These are notes for an introductory course on linear logic, written by non-logicians, and intended for non-logicians. Specifically, the notes are intended for linguists who would like to read more about some of the applications that linear logic has found in their field, but are put off by the logical background required. Most papers on linguistic applications of linear logic presuppose fairly extensive logical competence. For the average theoretical or computational linguist, existing texts on linear logic such as [Troelstra] provide a dauntingly technical introduction to this material. Our hope is to provide a more gentle and accessible introduction, focusing on those aspects of linear logic of most direct linguistic relevance.

As a prerequisite we assume some (perhaps fading) recollection of a standard introductory course on Montague semantics. That is, a reasonable familiarity with first-order predicate calculus, a rudimentary knowledge of the lambda-calculus and type theory, and a basic idea of what logical inference is about. If you have an intuitive understanding what the following mean:

$$\forall x.\ \mathsf{man}(x) \rightarrow \mathsf{mortal}(x),\quad \mathsf{man}(\mathsf{john}) \quad \vdash \quad \mathsf{mortal}(\mathsf{john})$$

$$\lambda x.\mathsf{see}(x, \mathsf{john})\ (\mathsf{fred}) \quad \equiv_\beta \quad \mathsf{see}(\mathsf{fred}, \mathsf{john})$$

then you probably meet the prerequisite.

Linguistic applications of linear logic have principally been in the areas of:

- Categorial and type-logical grammar [Moortgat,vanBenthem], including work on parsing categorial grammars [Morrill,Hepple], and the compositional semantics of categorial grammars [Morrill,Carpenter]

- ‘Glue semantics’, which to a first approximation is a version of categorial semantics but without an associated categorial grammar [Dalrymple]

- Resource-based reformulations of other grammatical theories, such as Minimalism [Retore,Stabler], Lexical Functional Grammar [Saraswat,Muskens] and Tree Adjoining Grammar [Abrusci]
- There have also been applications to such AI issues as the frame problem [White], which have some linguistic relevance

We reiterate that this course aims at giving the necessary *logical* background for beginning to understand the linguistic applications of linear logic. That is, our focus will be slanted more to logic than to linguistics. Rather than attempt to exhaustively describe all the linguistic applications listed above, we will single out only two for more sustained description: categorial grammar, and glue semantics. Others will receive much briefer mention. This approach may alienate a possible secondary audience for these notes, namely linear logicians who would like to learn something about linguistics. We hope that they will nonetheless find some things to interest them. But for our primary audience, linguists who would like to learn something about linear logic, we believe that the logical focus is likely to target the region of maximum need.

1.2 The Linguistic Appeals of Linear Logic

Linear logic has often been branded as a *resource-conscious* logic, or a logic of resources. This has undoubtedly been a major part of its linguistic attraction. Resource usage is an appealing metaphor for thinking about various linguistic issues. For example, how a string of words provides a sequence of resources that can be consumed to construct a syntactic analysis of a sentence. Or how word meanings provide a collection of resources that can be used to construct the meaning of a sentence. Or how linguistic context can make certain resources available, such as possible pronoun antecedents, that can be used to flesh out the interpretations of words like *he*, *she* or *it*. Indeed, it was this view of linguistic context as a consumable and updatable resource that originally attracted the present authors to the possible applications of linear logic.

But it would be a mistake to think that linear logic was originally devised for the purpose of being a logic of resources, in the way that e.g. tense logics were devised to be logics of time. Much of the initial motivation came from an altogether different direction: the role of *proofs* in logic. As Girard puts it [Gir:ssll] "linear logic comes from a proof-theoretic analysis of usual logic." To the extent that linear logic is a logic of resources, the resources in question are premises, assumptions and conclusions as they are used in logical proofs.

In brief, there is a programme to elevate the status of proofs to first class logical objects: instead of asking 'when is a formula A true', we ask 'what is a proof of A?'. Following Frege's distinction between sense and denotation, proofs are to constitute the senses of logical formulas, whose denotations might be truth values. But there is a problem with viewing proofs as logical objects. We do not have direct access to proofs, only to syntactic representations of them, in the form of derivations in some proof system (e.g. axiomatic, natural deduction or sequent calculus). As current proof theory stands, syntactic derivations are a flawed means of accessing the underlying proof objects. The syntax can introduce spurious differences between derivations that do not correspond to differences in the underlying proofs; it can also mask differences that really are there.

Some of the impetus behind the development of linear logic was to look more closely at proofs, and to obtain a more satisfactory way of describing the underlying proof objects. Indeed, one of the outcomes of linear logic is a new way of representing derivations, proof nets, that supposedly more accurately reflect the structure of underlying proofs.

It is not immediately obvious, perhaps, that this focus on proofs gives linear logic further linguistic appeal. But it does. We will point to just one example (which we expand on below): *parsing as deduction.* Under this view, parsing a sentence amounts to performing a logical proof in a system where the words of the sentence provide the premises, and the grammar the rules of inference. The parse tree of the sentence corresponds to the proof tree of the derivation. Moreover, this parse/proof tree is an object semantic significance: it can be used to construct the meaning of this sentence. From this perspective, it is natural to take proofs as first class objects, and to want to distinguish underlying proofs from the idiosyncracies of a particular way of representing derivations.

1.3 Resource Counting

In traditional logics formulas denote truths or facts. These facts may be used as little or as often as one likes. Take some simple facts about integers

$$\begin{array}{l} 5 > 4, \quad 4 > 3 \\ 1 > 0, \quad 2 > 0 \\ \forall i, j, k.\ (i > 0 \ \wedge \ j > k) \ \rightarrow \ (i \times j) > (i \times k) \end{array}$$

From these facts we can readily conclude, amongst other things, that

$$(1 \times 5) > (1 \times 4) \ \wedge \ (2 \times 5) > (2 \times 4)$$

Reaching this conclusion makes one use apiece of the facts that $1 > 0$ and $2 > 0$, and two uses apiece of $5 > 4$ and the universally quantified fact. It makes no use at all of the fact that $4 > 3$. The truth of $5 > 4$ does not wear out with repeated use: it remains just as true however many times we make use of it. Likewise, the truth of $4 > 3$ does not diminish into falsehood through lack of use: it remains just as true however few times we make use of it.

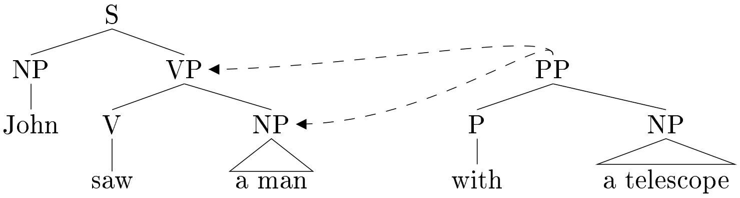
In linear logic formulas denote resources. Resources, unlike truths, get used up. A stock example of resource consumption is: (a) a packet of Gauloises costs 20FF, (b) a packet of Gitanes costs 20FF, therefore (c) if I have (a resource of) 20FF I can either buy a packet of Gauloises, or buy a packet of Gitanes, but not both.

Resource usage occurs in natural language at a very fundamental level. To a first approximation, each word and phrase in sentence is a resource that must get used exactly once in building a sentence. If a word needs to be used twice in building a sentence, it will occur in it twice[1]. And if a word is not to be used in building a sentence, it should not occur in it.

As a more concrete example, consider an ambiguous sentence like

> *John saw a man with a telescope.*

where the prepositional phrase (PP) "*with a telescope*" can either modify the noun phrase (NP) "*a man*" (so that the man has the telescope), or the verb phrase (VP) "it saw a man" (so that John is looking through the telescope). We can picture this state of affairs as follows



Here we have the phrase "*with a telescope*", which can attach either to an adjacent VP or to an adjacent NP. The crucial point is that it has to be one or the other, but not both. If the PP attaches to the NP, it gets used up and cannot be used a second time to attach to the VP; and vice versa. The sentence cannot be interpreted as saying that John looked through a telescope and saw a man holding a telescope.

As a slightly more formal illustration of the resource differences between traditional and linear logic, we can compare some valid and invalid patterns of inference. We will use the symbols $\rightarrow$ and $\wedge$ for traditional implication and conjunction, and $\multimap$ and $\otimes$ for linear implication and (multiplicative) conjunction.

First, premises get consumed in linear logic in a way that they don't in traditional logic:

Traditional implication:	$A, A \rightarrow B \vdash B$	
	$A, A \rightarrow B \vdash A \wedge B$	A is still true
Linear implication:	$A, A \multimap B \vdash B$	
	$A, A \multimap B \nvdash A \otimes B$	A is used up

In combining the two premises A and $A \multimap B$ to derive B, both the premises are used up. This means that there is no longer an A (nor an $A \multimap B$) around to be conjoined with the B. But in traditional logic, both premises are still available for re-use.

Unlike traditional logic, premises can't be ignored in linear logic

[1] Coordination is an exception to this crudely stated generalization. The sentence "*John ate and drank*" bears an equivalence to "*John ate and John drank*", where the word "*John*" occurs and is used twice. It is these occasional violations of a uniform 'use-exactly-once' regime that lends linear logic much of its interest. Linear logic, as we will see, permits quite fine-grained control over regimes for resource usage.

Traditional conjunction:	$A \wedge B \vdash A$	Can ignore B
Linear conjunction:	$A \otimes B \not\vdash A$	Have to use up B

In linear logic, if you have an A and a B resource, you cannot just throw one of them away; to get rid of it, you have to use it up. In traditional logic, you can always choose to ignore some truths.

We will meet further linear logic connectives besides $\multimap$ and $\otimes$ shortly, though the logical fragment defined by just these two is already of considerable linguistic importance. But one other connective that is worth mentioning early is the exponential or modality ! (variously pronounced: 'bang', 'pling', 'shriek', or 'of course'). The modality allows repeated use or discarding of any formula/resource to which it applies

Of course:	$!A \vdash A \otimes !A$	Re-use
	$!(A) \otimes B \vdash B$	Discard

Judicious use of this modality allows finer-grained control over resource sensitivity. At one extreme, by banging every formula you get traditional, non-resource sensitive logic.

1.4 Proofs as Objects

1.4.1 Structural Rules and Premise Counting

Linear logic, and its inherent resource sensitivity, arises from from consideration of the way that premises are used in proofs[2]. Traditionally, proofs are from (sub)sets of premises. The identity of a set is not changed if elements in it are repeated: hence premises can be re-used. If a conclusion follows from a certain set of premises, further premises can be added without invalidating the conclusion. Likewise, premises not used in a proof can safely be removed: premises can be spirited out of thin air, and discarded in the same way. Finally, the order of the premises in the set is immaterial to the identity of the set.

These observations about sets of premises point two three *structural* rules of inference at work in traditional logic. The first is contraction[3]:

$$\frac{\Gamma,\ A,\ A \vdash B}{\Gamma,\ A \vdash B}\ \mathit{contraction}$$

This says that if B follows from Γ plus two uses of A, it follows from Γ plus a single occurrence of A. We can always duplicate the A to get the second occurrence.

The second structural rule is weakening:

[2] Because of this it has recently been argued [Pym] that linear logic is a fairly impoverished logic of resources. It works well for premise counting, but not so well for more naturally occurring, divisible resources like time or money. Whatever the merits of this argument, the kind of resource sensitivity corresponding to premise counting is linguistically the most useful.

[3] For now, we are being fairly informal and sloppy in the way we present these rules, and the intuitive explanations are more important. More rigour will ensue when we discuss proof systems properly.

$$\frac{\Gamma \vdash B}{\Gamma,\ A \vdash B}\ \mathit{weakening}$$

This says that if B follows from Γ, it does no harm to expand, stretch or weaken the set of premises to include A.

The third structural rule is exchange:

$$\frac{\Gamma,\ A,\ B \vdash C}{\Gamma,\ B,\ A \vdash C}\ \mathit{exchange}$$

This says that the order in which the premises are presented does not matter.

Linear logic results from dropping the rules of contraction and weakening, so that premise counting becomes important. Rather than (sub)sets of premises, linear logic operates on *multisets* of premises. Dropping the two structural rules leads directly to the need to define new linear connectives, though we will defer discussion of this until a later chapter.

Linear logic preserves the rule of exchange however, so that premise order remains unimportant. For grammatical applications, premise order often corresponds to word order in a sentence, and is very important. Non-commutative linear logics result from modifying the exchange rule to account for order phenomena.

1.4.2 Proof Structures

The development of linear logic was also, in part, motivated by a deep interest in the structure of proofs. A major branch of proof theory is concerned not with just establishing whether a conclusion follows from its premises, but with the form any such proof takes. Independent of it specific application to linear logic, this concern with proof structure is of major significance to linguists; it should be particularly familiar to those operating within the paradigm of 'parsing as deduction'. We will therefore spend some time introducing the basic ideas for traditional (i.e. non-linear) logic, before briefly bringing the discussion back to linear logic.

Parsing as Deduction A version of (context free) parsing as deduction holds that grammar rules like

$$\begin{array}{llll}
\mathrm{S} & \Rightarrow & \mathrm{NP} & \mathrm{VP} \\
\mathrm{VP} & \Rightarrow & \mathrm{V} & \mathrm{NP} \\
\mathrm{PP} & \Rightarrow & \mathrm{P} & \mathrm{NP} \\
\mathrm{NP} & \Rightarrow & \mathrm{NP} & \mathrm{PP} \\
\mathrm{VP} & \Rightarrow & \mathrm{VP} & \mathrm{PP} \\
\mathrm{NP} & \Rightarrow & \mathrm{Det} & \mathrm{N}
\end{array}$$

should be regarded as logical implications in reverse; for example the existence of an adjacent Noun Phrase and Verb Phrase implies the existence of a Sentence spanning them both. Marking parameterized string positions (i, j, k) on the rules, we can rewrite them as logical implications

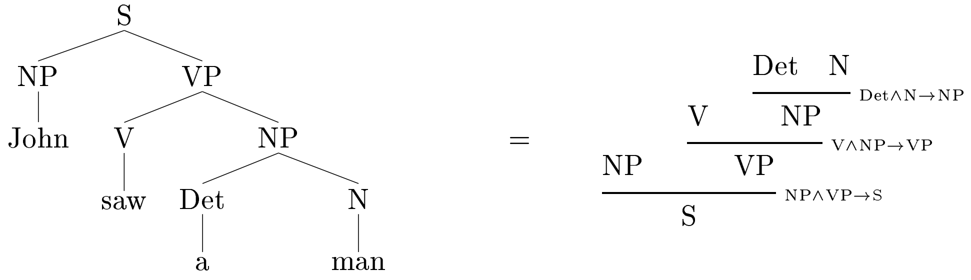
$$\begin{array}{llllll}
{}_i\mathrm{NP}_j & \wedge & {}_j\mathrm{VP}_k & \rightarrow & {}_i\mathrm{S}_k \\
{}_i\mathrm{V}_j & \wedge & {}_j\mathrm{VP}_k & \rightarrow & {}_i\mathrm{VP}_k \\
{}_i\mathrm{P}_j & \wedge & {}_j\mathrm{NP}_k & \rightarrow & {}_i\mathrm{PP}_k \\
{}_i\mathrm{NP}_j & \wedge & {}_j\mathrm{PP}_k & \rightarrow & {}_i\mathrm{NP}_k \\
{}_i\mathrm{VP}_j & \wedge & {}_j\mathrm{PP}_k & \rightarrow & {}_i\mathrm{VP}_k \\
{}_i\mathrm{Det}_j & \wedge & {}_j\mathrm{N}_k & \rightarrow & {}_i\mathrm{NP}_k
\end{array}$$

The first rule says that an NP from position $i - j$ and a VP from position $j - k$ implies a sentence from position $i - k$. A sentence like "*John saw a man with a telescope*" gives rise to a set of lexical premises

$$\begin{array}{lllll}
{}_0\mathrm{NP}_1, & {}_1\mathrm{V}_2, & {}_2\mathrm{Det}_3, & \ldots, & {}_6\mathrm{N}_7 \\
\text{John} & \text{saw} & \text{a} & \ldots & \text{telescope}
\end{array}$$

Letting Γ be the lexical premises plus the grammar rules, parsing becomes a search for a proof that $\Gamma \vdash {}_0\mathrm{S}_7$. What is interesting is not that ${}_0\mathrm{S}_7$ follows from Γ so much as there are two quite different ways of proving it. These two proofs correspond to the readings where (i) the PP "*with a telescope*" attaches to the NP "*a man*", and (ii) the PP attaches to the VP "*saw a man*"[4]. The bald statement that ${}_0\mathrm{S}_7$ follows from Γ does not reveal the existence of the two parses. Studying the structures of the proofs, on the other hand, does reveal the number of parses. Moreover, these differences in proof structure give rise to differences in meaning when we come to apply some form of compositional semantic interpretation.

More generally, for parsing as deduction proof structures *are* syntactic structures. A phrase structure tree can be viewed as a proof tree (though by convention, logicians write their proof trees upside down). For example



[4] The alert reader will have realized that without any restriction on resource usage there is a third proof: where the PP attaches to *both* the NP *and* the VP. Implementations of parsing as deduction, such as Direct Clause Grammars [???], usually include implicit resource control in the inference engine to prevent this. Using linear logic makes resource issues explicit in the (logicized) grammar, rather than implicit in its interpreter.

Furthermore, these structures carry semantically relevant information. In fact, proof structures have a non-trivial semantics, expressible by means of the lambda-calculus. This is not without significance for linguists working on the semantics of natural language.

The Semantics of Proofs — the Curry-Howard Isomorphism: Within proof theory, the 'semantics' of proof structures is a topic of major interest. It turns out that superficially distinct proofs can sometimes be semantically equivalent. These distinct but equivalent proofs can often be grouped together into an equivalence class, identifiable by a canonical form of the proof. Purely syntactic operations on any non-canonical proof, such as cut-elimination or proof normalization, can convert it to the canonical form. At a semantic level, these conversions amount to performing various types of lambda-reduction on the semantics of the proofs.

Although we will discuss the intimate connection between proofs, type-theory and the lambda calculus in more detail in chapter 2, a preliminary illustration is in order. Consider the standard natural deduction elimination and introduction rules for implication:

$$\frac{A \to B \qquad A}{B} \to_{\mathcal{E}} \qquad\qquad \frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{A \to B} \to_{\mathcal{I}}, i$$

The elimination rule $\to_{\mathcal{E}}$ is just *modus ponens*, from A and $A \to B$, conclude B. The introduction rule $\to_{\mathcal{I}}$ is read as follows. Assume that A, and label this assumption i for book-keeping purposes. Suppose that from the assumption A plus some other premises, you can prove B. Then you can discharge the assumption of A to instead conclude that if you were given A as a real premise you could derive B; i.e. conclude $A \to B$. The box around the assumption marks it as discharged, and the index on the introduction rule shows which assumption is being discharged. The conclusion $A \to B$ is no longer depedendent on the assumption A^i in the way that the intermediate conclusion B is.

Both of these standard inference rules can be paired with semantic operations on *(proof) terms*:

$$\frac{f : A \to B \qquad a : A}{f(a) : B} \to_{\mathcal{E}} \qquad\qquad \frac{\begin{array}{c}[x : A]^i \\ \vdots \\ f(x) : B\end{array}}{\lambda x.f(x) : A \to B} \to_{\mathcal{I}}, i$$

We should think about the rule for *modus ponens* / $\to_{\mathcal{E}}$ as follows: An implication $A \to B$ can be regarded as a function that takes things of type A and returns things of type B. Let us call the function f, and note that it will have a type $A \to B$. Suppose that we have an object a of type A. Applying the function f to it will give us an object $f(a)$, which has type B. Or put another way, the semantics of modus ponens correspdonds to the *functional application* of the implication to its (antecedent) argument.

The rule for implication introduction, $\rightarrow_{\mathcal{I}}$, corresponds to lambda-abstraction. Assuming some arbitrary x of type A, let us suppose we can construct some object $f(x)$ of type B. Then we can abstract over the arbitrary x to create a function $\lambda x.f(x)$ that can take any object of type A as an argument, and return an object of type B. That is, the function $\lambda x.f(x)$ has type $A \rightarrow B$.

The pairing of proof rules with (λ-calculus) operations on proof terms is known as the *Curry-Howard Isomorphism*[5] The term labelling each formula can be seen as a description of how the formula is derived/proved. (Premises are usually labelled with arbitrary atomic terms). Conversely, the formula, can be seen as giving the logical type of the proof.

Lambda-equivalences between proof terms can be used to show when two superficially distinct proofs are essentially the same. As an example here are two proofs that from A and $A \rightarrow B$ you can conclude B. The first proof (1a) is sensible, the second (1b) is pointlessly complicated:

(1) a.
$$\dfrac{A \rightarrow B \qquad A}{B}\rightarrow_{\mathcal{E}}$$

b.
$$\dfrac{A \qquad \dfrac{\dfrac{[A]^i \qquad A \rightarrow B}{B}\rightarrow_{\mathcal{E}}}{A \rightarrow B}\rightarrow_{\mathcal{I}}, i}{B}\rightarrow_{\mathcal{E}}$$

By considering proof terms we can show that the second, more complex derivation is an uninteresting variant of the first:

(2) a.
$$\dfrac{f : A \rightarrow B \qquad a : A}{f(a) : B}\rightarrow_{\mathcal{E}}$$

b.
$$\dfrac{a : A \qquad \dfrac{\dfrac{[x : A]^i \qquad f : A \rightarrow B}{f(x) : B}\rightarrow_{\mathcal{E}}}{\lambda x.f(x) : A \rightarrow B}\rightarrow_{\mathcal{I}}, i}{(\lambda x.f(x))(a) : B}\rightarrow_{\mathcal{E}}$$

Given the standard lambda-calculus rules of β- and η-conversion[6] note how the proof

[5] The isomorphism does not hold for all logics, and it does not hold for styles of proof system. The original version of the isomorphism was discovered for intuitionistic (as opposed to classical) logic, and for a natural deduction (as opposed to axiomatic) proof system.

[6] These rules are:

- β-conversion (lambda-reduction): $(\lambda x.\phi)(a) \;=\; \phi[a/x]$
  where $\phi[a/x]$ indicates substitution of x by a in ϕ
- η-conversion (extensionality): $\lambda x.\phi(x) \;=\; \phi$
  provided that x does not occur in ϕ

terms for (2a) and (2b) are equivalent,

$$f(a) \;=\; (\lambda x.f(x))(a)$$

indicating a semantic equivalence between the two derivations. Chapter 2 describes operations of *proof normalization*, corresponding to β- and η-conversion of proof terms, that reduce derivations to their simplest canonical form. These show, for example, that (1a) is the normal-form version of derivation (1b).

To sum up, proofs structures are interesting in part because they have non-trivial identity criteria. In well-behaved logical systems, superificially distinct proofs can be mapped onto a common, canonical form. Moreover, terms can be assigned to formulas in a proof, embuing the proof with some form of semantics. Proof normalization operations are meaning-preserving with regard to proofs. The relevance of this for linguists is twofold. If proof trees can correspond to parse trees, the existence of canonical form proofs suggests ways of limiting one's search for parse trees. Second, the semantics of proofs gives a handle on the semantics of parse trees. That is, the Curry-Howard Isomorphism can be a major tool for natural language semantics. This is indeed precisely the trick employed by categorial grammar as well as by glue semantics.

Linear Logic & Semantically Equivalent Proofs: Our discussion of the Curry-Howard Isomorphism, with its semantic identities between prooofs, has not touched on linear logic. However, as mentioned in section **??**, the structure and identity criteria for proofs is a major theme lying behind much work on linear logic. Not only can versions of the Curry-Howard Isomorphism be extended to linear logic, but new ways of representing certain types of proof are available — proof nets.

1.5 A Rough Guide to the Linear Connectives

In this section we give an informal introduction to the connectives and constants of linear logic, in the hope of providing the reader with an intuitive feel for their meanings. This task is not as easy as we would have liked. Some of the connectives and constants of linear logic are peculiarly resistant to informal explanation. Where this is the case, we will refer the reader to chapter 3 for explanation, and not attempt it here. It should be borne in mind that the explanations given here are meant to be highly informal, and solely for the purpose of providing the reader with some initial familiarisation.

One of the most immediately striking things about linear logic is that it has two forms of conjunction ($\otimes$ and &) and two forms of disjunction ($\parr$ and $\oplus$). To go with these, it also has two forms of true ($\top$ and $\mathbf{1}$), and two forms of false ($\perp$ and $\mathbf{0}$). Fortunately, there is just one form of implication, so we will start with this:

- Linear implication: $\multimap$
  $A \multimap B$ means that can consume an A resource to produce a B resource.

- Negation: $.^{\perp}$
  $A^{\perp}$ roughly speaking stands for something that will consume an A resource. Resources come paired, a little like matter and anti-matter. A production A meets with a consumption $A^{\perp}$ to leave nothing at all. Negation of a consumer gives rise to a producer, and vice versa, so that $A^{\perp\perp} \equiv A$.

- Tensor (multiplicative conjunction): $\otimes$
  $A \otimes B$ means that your resources make both A and B available. If you have a resource $A \otimes B$, you can recover both A and B simultaneously.

- Par (multiplicative disjunction): $\parr$
  This is one of the connectives that is hardest to explain. One way of looking at it is merely to note that $A \multimap B$ can be defined as $A^{\perp} \parr B$. That is either you have something that is looking to consume an A resource, or you produce a B resource. One can try to paraphrase $A \parr B$ as ‘if you don’t have an A then you have a B, and vice versa.’

- With (additive conjunction): &
  $A\&B$ means that your resources can make A available, and they can make B available, but not both simultaneously. In terms of proofs, your premises allow a proof establishing A, and they also allow a (separate) proof establishing B. But because proofs consume premises, you cannot put both of these proofs together using just the one set of premises.
  This is also sometimes known as *internal choice*: we can decide whether to obtain A or to obtain B.

- Plus (additive disjunction): $\oplus$
  $A \oplus B$ means your resources make either A or B available, but you don’t know which.
  This is also sometimes known as *external choice*

- Of course: !
  $!A$ means that you can produce as many copies of the A resource as you like, including zero copies.

- Why not: ?
  $?A$ means that you can consume as many copies of the A resource as you like, including zero copies.

- Unit: **1**
  This is the identity for tensor, so that $(A \otimes \mathbf{1}) \equiv A$. Unit is the trivial resource that can be produced from nothing. Another way of putting this is that if a collection of resources produces **1** (and nothing else), then we can consume / throw away that collection of resources

- Top: $\top$
  This is the identity for with, so that $(A\&\top) \equiv A$. Top consumes all resources

- Imposibility: $\mathbf{0}$
  This is the identity for plus, so that $(A \oplus \mathbf{0}) \equiv A$. It corresonds to the impossible resource, so that an external choice between A and $\mathbf{0}$ must always result in A. Note also that $\mathbf{0} \equiv \top^{\perp}$. Since $\top$ consumes all resources, $\mathbf{0}$ produces all resources. In this respect, it is like logical falsehood, from which all possible conclusions follow.

- Bottom: $\perp$
  This is the identity for par, so that $(A \parr \perp) \equiv A$. It is also the dual of $\mathbf{1}$, so that $\mathbf{1}^{\perp} \equiv \perp$

This is a large collection of connectives. The reader will be cheered to know that for linguistic purposes implication and tensor are by far and away the most significant.

It seems to have become a tradition to illustrate the meanings of some of the connectives in relation to the interpretation of a fixed price menu at a restaurant. Not wishing to break the tradition:

Menu: £5	$(P \otimes P \otimes P \otimes P \otimes P)$
	$\multimap$
Fish	$[Fish$
	$\otimes$
Chips	$Chips$
	$\otimes$
Soup or Salad	$(Soup\&Salad)$
	$\otimes$
Fruit or cheese (depending on availability)	$(Fruit \oplus Cheese)$
	$\otimes$
Coffee (free refills)	$Coffee$ $!Coffee]$

Here we begin with five one pound resources (P). Note how & is used to mark the choice over which we have control (soup or salad), whereas $\oplus$ is used to mark the choice over which the restaurant has control (fruit or cheese). The implication could also be represented as

$$(P \otimes P \otimes P \otimes P \otimes P)^{\perp} \parr [Fish \otimes \ldots \otimes !Coffee]$$

This says that either we are left with something that wants to consume five pounds, or we are left with the meal.

Chapter 2

Basic Proof Theory

This chapter reviews some basic aspects of proof theory, but does not touch on linear logic. It is here to set the scene for the next chapter, which gives a more formal introduction to linear logic, along with some of its proof theoretic motivations.

There are three main ways of formulating proof systems for logics: axiomatic, natural deduction, and sequent calculus. We will have little to say about axiomatic systems. We will focus mostly on proof systems for propositional logic, and will distinguish between classical and intuitionistic logics. We will also talk about the ways in which proofs can be normalized to canonical forms, and the Curry-Howard isomorphism.

2.1 Natural Deduction

Natural deduction proof systems are characterised by giving paired inference rules for logical connectives: an *introduction* rule showing how to introduce an instance of the connective into a derivation, and an *elimination* rule showing how to remove an instance of the connective from a derivation. Natural deduction was originally devised to try and reflect typical practice in carrying out logical proofs. Moreover, the introduction and elimination rules lend themselves to an intuitive understanding of the meaning and use of the connectives they define.

2.1.1 ND for Classical Propositional Logic

Conjunction As a first illustration of natural deduction proof rules, let us consider the case of conjunction. Intuitively, the introduction rule should behave as follows. Suppose, in the course of a proof, you have derived A and you have also derived B. Then you should be able to put these two sub-derivations together to derive $A \wedge B$.

We can represent this pattern as

$$\dfrac{\begin{matrix}\vdots & \qquad & \vdots\\ A & \qquad & B\end{matrix}}{A\wedge B}$$

The dots above A and B represent derivations resulting in A and B. We will generally omit these dots in what follows, giving the rule of "And-Introduction", $\wedge_{\mathcal{I}}$

$$\dfrac{A \qquad B}{A\wedge B}\wedge_{\mathcal{I}}$$

To eliminate a conjunction $A\wedge B$, we can choose to drop either one of the conjuncts. This gives rise to two mirror image elimination rules

$$\dfrac{A\wedge B}{A}\wedge_{\mathcal{E}} \qquad\qquad \dfrac{A\wedge B}{B}\wedge_{\mathcal{E}}$$

Implication The elimination rule for implication is familiar as *modus ponens*

$$\dfrac{A \qquad A\to B}{B}\to_{\mathcal{E}}$$

The rule for introducing an implication $A\to B$ is slightly more involved. Intuitively, it works as follows. Assume, for the sake of argument, that A holds. Suppose that from this assumption, plus whatever other premises you have, that you can conclude B. Then you can *discharge* the assumption A to conclude that "if A were to hold, then B would hold." The formal rule is:

$$\dfrac{\begin{matrix}[A]^i\\ \vdots\\ B\end{matrix}}{A\to B}\to_{\mathcal{I},i}$$

There are a few things to note about this rule. First, we use

$$\begin{matrix}A\\ \vdots\\ B\end{matrix}$$

to represent a derivation of B starting from A plus any other premises available. Second, we use an (arbitrary) index i to identify the assumption of A. This is shown as a superscript on the assumption, and also alongside the $\to_{\mathcal{I}}$ rule that discharges the assumption. Finally, the assumption is enclosed in square brackets to show that it has been discharged by the $\to_{\mathcal{I},i}$ rule (an alternative is to draw a slash through the discharged assumption).

Here is an example derivation to illustrate how these rules work

$$\dfrac{\dfrac{\dfrac{(A \land B) \to C \qquad \dfrac{[A]^1 \quad [B]^2}{A \land B}\,\land_{\mathcal{I}}}{C}\,\to_{\mathcal{E}}}{B \to C}\,\to_{\mathcal{I},2}}{A \to (B \to C)}\,\to_{\mathcal{I},1}$$

This derivation shows that $A \to (B \to C)$ can be derived from $(A \land B) \to C$.

Notes on Discharging Assumptions:

1. A premise, like $(A \land B) \to C$ in the derivation above, is simply an assumption that is not discharged. To avoid clutter, we do not index such undischarged assumptions.
2. Multiple assumptions (of the same formula) are allowed to have the same index. The elimination rule for the index discharges *all* of these assumptions. Here is an example of a derivation with multiple occurrences of an assumption

$$\dfrac{\dfrac{\dfrac{A \to (B \to C) \qquad \dfrac{[A \land B]^1}{A}\,\land_{\mathcal{E}}}{B \to C}\,\to_{\mathcal{E}} \qquad \dfrac{[A \land B]^1}{B}\,\land_{\mathcal{E}}}{C}\,\to_{\mathcal{E}}}{(A \land B) \to C}\,\to_{\mathcal{I},1}$$

3. It is also possible to discharge *non-existent* assumptions. For example

$$\dfrac{\dfrac{[A]^1}{B \to A}\,\to_{\mathcal{I},2}}{A \to (B \to A)}\,\to_{\mathcal{I},1}$$

Note that this is also a derivation with no premises / undischarged assumptions. The conclusion, $A \to (B \to A)$ is therefore a *theorem* of the logic

Disjunction There are two mirror image rules for introducing a disjunction (just as there are for eliminating a conjunction)

$$\dfrac{A}{A \lor B}\,\lor_{\mathcal{I}} \qquad\qquad \dfrac{A}{B \lor A}\,\lor_{\mathcal{I}}$$

This allows us to introduce an arbitrary disjunct, B, so long as we know that A holds.

The elimination rule is cumbersome (Girard sometimes describes it as the "shame of natural deduction"). To eliminate $A \lor B$, we know that one or other of A and B hold (and possibly both), but we don't know which. We can replace $A \vee B$ by C if C follows from A and C also follows from B. That is, whichever one of A or B turns out to be the one that holds, we can be sure that C is derivable. This is sometimes known as 'proof-by-cases'. The rule is:

$$\dfrac{A \vee B \qquad \begin{matrix}[A]^i \\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]^j \\ \vdots \\ C\end{matrix}}{C}\ \vee_{\mathcal{E}i,j}$$

Another unpleasant feature of this rule that the conclusion, C, is a formula that has nothing to do with the formula being eliminated $A \vee B$. The conclusion is sometimes referred to as a *parasitic* formula.

Negation There are a number of different ways of handling negation. In fact, the difference between classical and intuitionistic logic can be captured through the treatment of negation.

The most compact treatment of negation is to introduce a constant $\perp$ (pronounced *falsum*), which represents falsity. We can then define negation as implication into falsity:

$$\neg A \overset{df}{=} A \rightarrow \perp$$

We then have just *one* inference rule, known as *reductio ad absurdum*

$$\dfrac{\begin{matrix}[A \rightarrow \perp]^i \\ \vdots \\ \perp\end{matrix}}{A}\ \mathrm{RAA}_i$$

A notational alternative to introduce negation as a primitive connective, whose introduction and elimination rules are simply instances of $\rightarrow_{\mathcal{E}}$ and $\rightarrow_{\mathcal{I}}$ for $A \rightarrow \perp$

$$\dfrac{\begin{matrix}[A]^i \\ \vdots \\ \perp\end{matrix}}{\neg A}\ \neg_{\mathcal{I},i} \qquad\qquad \dfrac{A \qquad \neg A}{\perp}\ \neg_{\mathcal{E}}$$

The rule of *reductio ad absurdum* could then be reformulated as

$$\dfrac{\begin{matrix}[\neg A]^i \\ \vdots \\ \perp\end{matrix}}{A}\ \mathrm{RAA}_i$$

Introduction

$$\frac{\begin{matrix}[A]^i\\ \vdots \\ B\end{matrix}}{A \to B}\,\to_{\mathcal{I},i}$$

$$\frac{A \qquad B}{A \wedge B}\,\wedge_{\mathcal{I}}$$

$$\frac{A}{A \vee B}\,\vee_{\mathcal{I}} \qquad \frac{A}{B \vee A}\,\vee_{\mathcal{I}}$$

$$\frac{\begin{matrix}[A]^i\\ \vdots \\ \bot\end{matrix}}{\neg A}\,\neg_{\mathcal{I},i}$$

Elimination

$$\frac{A \qquad A \to B}{B}\,\to_{\mathcal{E}}$$

$$\frac{A \wedge B}{A}\,\wedge_{\mathcal{E}} \qquad \frac{A \wedge B}{B}\,\wedge_{\mathcal{E}}$$

$$\frac{A \vee B \qquad \begin{matrix}[A]^i\\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]^j\\ \vdots \\ C\end{matrix}}{C}\,\vee_{\mathcal{E}i,j}$$

$$\frac{A \qquad \neg A}{\bot}\,\neg_{\mathcal{E}}$$

$$\frac{\begin{matrix}[\neg A]^i\\ \vdots \\ \bot\end{matrix}}{A}\,\mathrm{RAA}_i$$

Figure 2.1: Natural Deduction for Classical Propositional Logic

A Note on $\bot$ The constant $\bot$ is a *unit* for disjunction. That is

$$A \vee \bot \equiv A$$

There is also another unit, $\top$ (or *verum*) for conjunction, such that

$$A \wedge \top \equiv A$$

These units can be likened to zero and one in arithmetic, where $N+0 = N$ and $N \times 1 = N$.

Summary The natural deduction system for classical propositional logic is summarised in figure 2.1.

Introduction

$$\dfrac{\begin{array}{c}[A]^i\\ \vdots\\ B\end{array}}{A \to B}\,{\to_{\mathcal{I},i}}$$

$$\dfrac{A \qquad B}{A \wedge B}\,{\wedge_{\mathcal{I}}}$$

$$\dfrac{A}{A \vee B}\,{\vee_{\mathcal{I}}} \qquad \dfrac{A}{B \vee A}\,{\vee_{\mathcal{I}}}$$

$$\dfrac{\begin{array}{c}[A]^i\\ \vdots\\ \bot\end{array}}{\neg A}\,{\neg_{\mathcal{I},i}}$$

Elimination

$$\dfrac{A \qquad A \to B}{B}\,{\to_{\mathcal{E}}}$$

$$\dfrac{A \wedge B}{A}\,{\wedge_{\mathcal{E}}} \qquad \dfrac{A \wedge B}{B}\,{\wedge_{\mathcal{E}}}$$

$$\dfrac{A \vee B \qquad \begin{array}{c}[A]^i\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[B]^j\\ \vdots\\ C\end{array}}{C}\,{\vee_{\mathcal{E}i,j}}$$

$$\dfrac{A \qquad \neg A}{\bot}\,{\neg_{\mathcal{E}}}$$

$$\dfrac{\bot}{A}\,{\bot_{\mathcal{E}}}$$

Figure 2.2: Natural Deduction for Intuitionistic Propositional Logic

2.1.2 ND for Intuitionistic Propositional Logic

The natural deduction system for intuitionistic propositional logic may be obtained by a slight variation on the system for classical logic. Remove the rule of *reductio ad absurdum*, and replace it by:

$$\dfrac{\bot}{A}\,{\bot_{\mathcal{E}}}$$

(an instance of RAA where a null assumption is discharged). The full system is shown in figure 2.2. Behind this apparently small change lies a wealth of difference.

Double Negation and the Excluded Middle First, note that the rule $\bot_{\mathcal{E}}$ is in fact a special case of the *reductio ad absurdum* rule for classical logic, where only a null assumption may be discharged. This means that intuitionistic logic proves fewer theorems than classical logic. In particular, the following two classical proofs rely on RAA and are not intuitionistically valid:

Proof of $\neg\neg A \to A$

$$\dfrac{\dfrac{\dfrac{[\neg\neg A]^1 \qquad [\neg A]^2}{\bot}\,\neg_{\mathcal{E}}}{A}\,\text{RAA}_2}{\neg\neg A \to A}\,\to_{\mathcal{I},1}$$

Proof of $A \vee \neg A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[(A\vee\neg A)\to\bot]^1 \qquad \dfrac{[A]^2}{A\vee\neg A}\,\vee_{\mathcal{I}}}{\bot}\,\to_{\mathcal{E}}}{\neg A}\,\neg_{\mathcal{I},2}}{A\vee\neg A}\,\vee_{\mathcal{I}} \qquad [(A\vee\neg A)\to\bot]^1}{\bot}\,\to_{\mathcal{E}}}{A\vee\neg A}\,\text{RAA}_1$$

The non-equivalence of A and $\neg\neg A$[1] and the non-validity of $A \vee \neg A$ are two of the hallmarks of intuitionistic logic, and illustrate its *constructive* nature.

Constructivism demands that whenever we show that something holds, we give an example. Thus, to show that there are prime numbers below 10, we should provide an example (e.g. 3) alongside a proof that the example is prime.

The entailment $\neg\neg A \to A$ does not lend itself to constructive examples. The fact that we have a proof that we cannot prove $\neg A$ does not immediately tell us what a positive proof of A would look like. This is even more apparent with the 'law of the excluded middle' $A \vee \neg A$. Since this can apply to any A whatsoever, it can tell us nothing about what a proof of a particular A (or $\neg A$) would look like.

As an example [Dalen] consider whether the decimal expansion of π contains nine 9s in a sequence. Classically, either it does or it doesn't. But this classically true statement gives us no clue as to how to begin establishing the matter one way or the other.

The Brouwer-Heyting-Kolmogorov (BHK) Interpretation Intuitionistic propositional logic cannot be given a truth table semantics in the way that classical logic can. However, it can be given a (constructive) semantics in terms of proofs that is not readily open to classical logic. This is the Brouwer-Heyting-Kolmogorov (BHK) Interpretation.

[1] Note that $A \to \neg\neg A$ *is* intuitionistically valid, even though $\neg\neg A \to A$ is not.

Suppose that each atomic formula q is paired with the proof Q that it holds (this might just be the bare fact that q). The proof Q gives the meaning of q. We now construct the proofs / meanings of complex formulas as follows

- P is a proof of $\phi \wedge \psi$ iff $P = \langle P_1, P_2 \rangle$ where P_1 is a proof of ϕ and P_2 is a proof of ψ
- P is a proof of $\phi \vee \psi$ iff $P = \langle LR, P_1 \rangle$ such that either $LR = 0$ and P_1 is a proof of ϕ or $LR = 1$ and P_1 is a proof of ψ.
  (That is P is a proof either of ϕ or ψ, plus an indication of which one it is a proof of.)
- P is a proof of $\phi \rightarrow \psi$ iff P is a construction / function that converts each proof P_1 of ϕ into a proof $P(P_1)$ of ψ.
- Nothing is a proof of $\bot$.

It is natural to ask why a similar interpretation cannot be given for classical logic. To see why not, we should consider the differences between the classical rule of RAA, and the intuitionistic rule of $\bot_{\mathcal{E}}$

$$\dfrac{\begin{array}{c}[\neg A]^i\\ \vdots\\ \bot\end{array}}{A}\,\mathrm{RAA}_i \qquad\qquad \dfrac{\bot}{A}\,\bot_{\mathcal{E}}$$

The rule of $\bot$ elimination is in some sense an inferential dead-end[2]. As soon as one derives $\bot$, one should give up that part of the derivation. If you're feeling perverse, you can extend the non-proof of $\bot$ into the same non-proof of A. But if you use this derivation of A any further, it will turn anything else it touches into a non-proof.

The situation is very different with *reductio ad absurdum*. The whole point there is to transform the non-proof of $\bot$ from $\neg A$ into a proof of A that can safely be used elsewhere. A difficulty that must be overcome in providing a BHK-style interpretation for classical logic is what the proof of A should look like. At the very least, it needs to take a non-proof and transform it into a proof. But we are in trouble if we take the (fairly reasonable) stance of identifying all non-proofs, in the same way that all empty sets, or all instances of the truth value false are identified. For then, we cannot distinguish between a RAA proof of A and B.

As we will see when discussing the Curry-Howard Isomorphism, the BHK interpretation is closely related to the fact that one can naturally associate lambda-terms with intuitionistic formulas, but not with classical formulas.[3]

[2] Indeed, minimal logic does without the inference rule altogether; it is just the classical/intuitionsitic rules for $\wedge, \vee, \rightarrow$ and $\neg$, but without either RAA or $\bot_{\mathcal{E}}$.

[3] There has been recent work on attempting to give a Curry-Howard Isomorphism for classical logic [[Parigot]]. The point is that it is much harder to do this for classical logic than intuitionistic logic.

Symmetry in Natural Deduction Intuitionistic logic also leads to a more symmetric natural deduction system. Recall that we said that natural deduction is characterised by paired introduction and elimination rules for connectives. The observent reader will have notice that the rule for RAA is not paired with anything other rule. And depending on the notational convention used, it either eliminates a negation ($\neg$) or implication ($\to$). That is, classical logic breaks the symmetry of natural deduction.

The symmetry is not broken by the rule $\bot_{\mathcal{E}}$. Rather than define a connective, this rule defines a constant. For constants, one expects to see either an elimination rule, or an introduction rule, but not both. (In fact, the constant $\bot$ has a dual $\top$ with the introduction rule

$$\frac{}{\top}\,\top_{\mathcal{I}}$$

Adding this rule does nothing to extend the power of intuitionistic (or classical) logic). As a consequence, the introduction and elimination rules for the connectives in intuitionistic logic are exactly and symmetrically paired.

Constructivism and Linear Logic We have mentioned the differences between classical and intuitionistic logic here because the issue is of relevance to linear logic. One of the aims of linear logic is to give a classical logic, where $\neg\neg A \equiv A$, but which is nonetheless constructive. Discussion of this will be deferred to the next chapter, however.

2.1.3 ND for Quantifiers

For the sake of completeness, we will give the natural deduction rules for quantifiers. These are the same rules for both classical and intuitionistic logic, though their different settings lend the quantifiers subtly different meanings.

$$\frac{A}{\forall x.A[x/a]}\,\forall_{\mathcal{I}}$$

Provided a does not occur in
any assumptions A depends on

$$\frac{\forall x.A}{A[a/x]}\,\forall_{\mathcal{E}}$$

$$\frac{A}{\exists x.A[x/a]}\,\exists_{\mathcal{I}}$$

$$\frac{\exists x.A \qquad \begin{array}{c}[A[x/a]]^i\\ \vdots\\ B\end{array}}{B}\,\exists_{\mathcal{E},i}$$

Provided a does not occur in $\exists x.A$
B or any assumption on which B depends
(except for $A[x/a]$)

The notation $A[x/a]$ denotes a uniform substitution of the term x for a throught A.

The intuitionistic quantifiers can be thought of as follows, under the BHK interpretation

- P is a proof of $\exists x.\phi$ iff P is a pair $\langle a, P_1\rangle$ such that P_1 is a proof of $\phi[a/x]$.
- P is a proof of $\forall x.\phi$ iff P is a construction that will take any object a and convert it into a proof $P(a)$ of $\phi[a/x]$.

2.1.4 Sequent Representations of Natural Deduction

Natural deduction derivations are trees of the form

$$\begin{array}{ccc} A_1 & \ldots & A_n \\ & \vdots & \\ & B & \end{array}$$

We can abbreviate this as a sequent

$$A_1, \ldots, A_n \vdash B$$

More generally, we can write

$$\Gamma \vdash B$$

where Γ stands for a set of assumptions / premises.

Given this abbreviation, we can rewrite the natural deduction rules as shown in figure 2.3

2.2 Sequent Calculus

A somewhat different proof system using sequent notation is the *sequent calculus*. We generalise the notion of sequent slightly, so that sequents are of the form

$$\Gamma \vdash \Delta$$

where both Γ and Δ are multisets of formulas. (Note that the sequent version of natural deduction used sequents where Δ was a single formula). A sequent

$$\gamma_1, \ldots, \gamma_i \vdash \delta_1, \ldots, \delta_j$$

means that "if the *conjunction* of $\gamma_1, \ldots, \gamma_i$ holds, then the *disjunction* of $\delta_1, \ldots, \delta_j$ holds." That is, the commas have different meanings dedending on which side of the turnstile ($\vdash$) they occur on. On the left hand side they correspond to conjunctions, and on the right side to disjunctions. Or put another way, a sequent $\Gamma \vdash \Delta$ means that if all of Γ holds, then at least one of Δ holds.

As with natural deduction, sequent rules for the connectives come in pairs. But here the pairs are to introduce the connectives on either the *left* or the *right* of the turnstile. There is a rough correlation between left-rules and elimination-rules, and right-rules and

$$\frac{\Gamma, A^i \vdash B}{A \to B} \to_{\mathcal{I},i} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash A \to B}{\Gamma \vdash B} \to_{\mathcal{E}}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_{\mathcal{I}} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge_{\mathcal{E}} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge_{\mathcal{E}}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_{\mathcal{I}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash B \vee A} \vee_{\mathcal{I}} \qquad \frac{\Gamma \vdash A \vee B \qquad \Gamma, A^i \vdash C \qquad \Gamma, B^j \vdash C}{\Gamma \vdash C} \vee_{\mathcal{E}i,j}$$

$$\frac{\Gamma, A^i \vdash \bot}{\Gamma \vdash \neg A} \neg_{\mathcal{I},i} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash \bot} \neg_{\mathcal{E}}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A} \bot_{\mathcal{E}} \qquad \frac{\Gamma, \neg A^i \vdash \bot}{\Gamma \vdash A} \mathrm{RAA}_i$$

$$\frac{}{\Gamma, A \vdash A} \, \mathit{axiom}$$

Figure 2.3: Sequent style ND Rules (Classical & Intuitionistic)

introduction-rules. However, this connection should not be pushed too far. There are also structural rules to ensure that the Γs and Δs really do behave as multi-sets.

For people who are familiar only with natural deduction proof systems, sequent calculus can be confusing at first. There is a tendency to try and read sequent proofs like natural deductions proofs. In some cases this can work, but more often it leads one astray. We will therefore spend a little while explaining how one should do proofs in sequent calculus — a piece of folklore not often explicitly described in textbooks. But first, we will present an actual proof system.

2.2.1 SC for Classical Propositional Logic

The sequent calculus proof system for classical propositional logic is shown in figure 2.4. Let us consider the rules in more detail.

Structural Rules The weakening rules allow one to add more formulas either to the left or to the right

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,\textit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\,\textit{Weakening}_{\mathcal{R}}$$

Weakening on the left means that if Γ proves Δ, then Γ and A still proves Δ. On the right, if Γ proves Δ, then Γ also proves Δ or A.

Contraction allows one to remove duplicated formulas on either the left or the right.

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\,\textit{Contraction}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\,\textit{Contraction}_{\mathcal{R}}$$

On the left, if Γ plus two As proves Δ, then so does Γ plus one A. On the right, if Γ proves Δ or A or A, then it also proves Δ or A.

Gentzen's original formulation of the sequent calculus treated the Γs and Δs are sequences (ordered lists). He therefore included additional structural rules to allow all permutations of of the sequences

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}\,\textit{Exchange}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta, B, A, \Delta_2}\,\textit{Exchange}_{\mathcal{R}}$$

These rules are unnecessary if Γ and Δ are already taken to be multisets.

Conjunction The left rule for conjunction is reminiscent of an elimination rule:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,\wedge_{\mathcal{L}}$$

Structural Rules

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,\textit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\,\textit{Weakening}_{\mathcal{R}}$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\,\textit{Contraction}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\,\textit{Contraction}_{\mathcal{R}}$$

Left-Right Rules

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,\wedge_{\mathcal{L}} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,\wedge_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\,\wedge_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta,}\,\vee_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B}\,\vee_{\mathcal{R}} \quad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}\,\vee_{\mathcal{R}}$$

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\,\to_{\mathcal{L}} \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B}\,\to_{\mathcal{R}}$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\,\neg_{\mathcal{L}} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\,\neg_{\mathcal{R}}$$

Axiom and Cut

$$\frac{}{A \vdash A}\,\textit{axiom}$$

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,\textit{cut}$$

Figure 2.4: Sequent Calculus for Classical Propositional Logic

If Γ and A proves Δ, then so will Γ and $A \wedge B$ — just ignore/eliminate the B conjunct. The right rule is even more reminiscent of an introduction:

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B} \wedge_{\mathcal{R}}$$

If Γ proves Δ or A, and it also proves Δ or B, then we can combine the A and B to conclude that Γ proves Δ or $A \wedge B$.

Note that while setting $\Delta = \emptyset$ in $\wedge_{\mathcal{R}}$ gives us exactly the sequent version of $\wedge_{\mathcal{I}}$, the connection between $\wedge_{\mathcal{L}}$ and $\wedge_{\mathcal{E}}$ is considerably more indirect. It probably hinders as much as it helps to read sequent rules as natural deduction rules.

Disjunction The left rule for disjunction carries much the same import as the convoluted rule for $\vee_{\mathcal{E}}$:

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma, A \vee B \vdash \Delta,} \vee_{\mathcal{L}}$$

If Γ and A proves Δ, and Γ and B also proves Δ, then Γ and $A \vee B$ will prove Δ — whichever disjunct, A or B, we pick, we can still derive Δ. The right rule $\vee_{\mathcal{R}}$ is identical to $\vee_{\mathcal{I}}$ when $\Delta = \emptyset$.

Implication The rules for implication are a little hard to decipher in the classical case (though they look a lot more familiar for intuitionistic logic). The left rule captures the effect of *modus ponens*, albeit in a very roundabout way

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta} \rightarrow_{\mathcal{L}}$$

Consider $\Gamma \vdash \Delta, A$, which says that Γ either proves Δ or it proves A. If Γ proves Δ, then by weakening it will be the case that Γ and $A \rightarrow B$ proves Δ. This is the uninteresting case. Now consider the case where Γ does not prove Δ, but only A. The second sequent, $\Gamma, B \vdash \Delta$ says that even if Γ does not prove Δ, Γ plus B will. And we also know that Γ proves A. Therefore, if we add $A \rightarrow B$ to Γ, we will in then be able to prove B, which will then allow us to prove Δ.

The right rule for implicationis very similar to $\rightarrow_{\mathcal{I}}$, except without the use of assumptions and associated book-keeping

$$\frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \rightarrow B} \rightarrow_{\mathcal{R}}$$

If we set Δ to $\emptyset$, the rules says that if Γ and (assumption) A prove B, then Γ on its own will prove $A \rightarrow B$.

Negation The two negation rules allow a formula to move across the turnstile, negating it as it goes.

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\,\neg_{\mathcal{L}} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\,\neg_{\mathcal{R}}$$

It is worth noting that it is only the $\neg_{\mathcal{R}}$ and *Weakening*$_{\mathcal{R}}$ rules that can increase the number of formulas on the right hand side of sequents. This is significant when considering intuitionistic logic.

The negation rules also lie behind the use of *one-sided* sequents. Any two sided sequent

$$\gamma_1, \ldots, \gamma_i \vdash \delta_1, \ldots, \delta_j$$

is equivalent to a one-sided sequent

$$\vdash \neg\gamma_1, \ldots, \neg\gamma_i, \delta_1, \ldots, \delta_j$$

One sided sequents are sometimes used as an alternative notation for the sequent calculus.

Cut The cut rule is one that is eliminable in all well-behaved sequent calculi (i.e. systems satisfying 'cut-elimination')

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,\mathit{cut}$$

It gives us as way of sticking or cutting together spearate sub-derivations. As with implication, the rule is easier to understand in the intuitionistic case. Here, we can understand it as follows: Case (i) suppose $\Gamma \vdash \Delta$. Then by weakening on both left and right $\Gamma, \Gamma' \vdash \Delta, \Delta'$. Case (ii) is the interesting one. Suppose $\Gamma \vdash A$. Then given that $\Gamma', A \vdash \Delta'$, we can cut the Γ in in place of A to give $\Gamma', \Gamma \vdash \Delta'$. This can be weakened to give $\Gamma, \Gamma' \vdash \Delta, \Delta'$.

In systems satisfying cut-elimination, proofs employing cut can be replaced by shorter proofs not using the cut rule. This is discussed in section 2.3.

Symmetry The sequent calculus formulation of classical logic is more symmetric than the natural deduction formulation.

2.2.2 SC for Intuitionistic Propositional Logic

The sequent calculus for intuitionistic logic may be obtained by restricting right hand sides of sequents to multisets of zero or one formulas. Sequents with an empty right hand side

$$\Gamma \vdash$$

can be thought of as

$$\Gamma \vdash \bot$$

The restriction on right hand sides necessitates reformulations of some of the rules, as shown in figure 2.5.

The restriction to single conclusion sequents substantially modifies the rule for negation on the right (and weakening on the right). This blocks the classical derivations of the following two central non-theorems of intuitionistic logic:[4]

Classical proof of $\neg\neg A \to A$:

$$\cfrac{\cfrac{\cfrac{A \vdash A}{\vdash \neg A, A}\,\neg_{\mathcal{R}C}}{\neg\neg A \vdash A}\,\neg_{\mathcal{L}}}{\neg\neg A \to A}\,\to_{\mathcal{R}}$$

Classical proof of $A \vee \neg A$:

$$\cfrac{\cfrac{\cfrac{\cfrac{A \vdash A}{\vdash \neg A, A}\,\neg_{\mathcal{R}C}}{A \vdash \vee\neg A, A}\,\vee_{\mathcal{R}}}{\vdash A \vee \neg A, A \vee \neg A}\,\vee_{\mathcal{R}}}{\vdash A \vee \neg A}\,\mathit{Contraction}_{\mathcal{R}}$$

Single and Multiple Conclusion Sequents Why does the restriction to single conclusion sequents furnish intuitionistic logic?[5] We will defer proper discussion of this question until later.

2.2.3 How to do Proofs in Sequent Calculus

A problem that many newcomers face is how to do proofs in the sequent calculus.[6] Sequent calculus is well suited for systematic, bottom up searches for derivations. This is often expressed as saying that sequent rules *should be read "upwards"*. The stanard pattern of application is to begin with the sequent you want to prove, and use the rules to work upwards to axioms. This is the reverse of the more top-down approach in natural deduction.

[4] A third non-intuitionistic classical theorem, Pierce's Law $((A \to B) \to A) \to A$, relies on the multiple conclusion version of $\mathit{Weakening}_{\mathcal{R}}$. So it is not just $\neg_{\mathcal{R}}$ that makes the substantial difference between classical and intuitionistic logic.

[5] An immediate caveat is to note that there can be multiple conclusion formulations of intuitionistic logic [[Dummett,Paiva]], though these require modifications elsewehere.

[6] It is not necessary to know how to do sequent proofs to understand the rest of these notes. This subsection is included because it is nonetheless a useful skill that is rarely described explicitly in text books.

Structural Rules

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \mathit{Weakening}_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash}{\Gamma \vdash B}\ \mathit{Weakening}_{\mathcal{R}}$$

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ \mathit{Contraction}$$

Left-Right Rules

$$\frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\wedge_{\mathcal{L}} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}\wedge_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\wedge_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}\vee_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\vee_{\mathcal{R}} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\vee_{\mathcal{R}}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C}\rightarrow_{\mathcal{L}} \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}\rightarrow_{\mathcal{R}}$$

$$\frac{\Gamma \vdash A}{\Gamma, \neg A \vdash}\neg_{\mathcal{L}} \qquad\qquad \frac{\Gamma, A \vdash}{\Gamma \vdash \neg A}\neg_{\mathcal{R}}$$

Cut and Axiom

$$\frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ \mathit{cut}$$

$$\frac{}{A \vdash A}\ \mathit{axiom}$$

Figure 2.5: Sequent Calculus for Intuitionistic Propositional Logic

As an example, let us consider how to go about proving the sequent

$$A \to (B \to C) \vdash (A \wedge B) \to C$$

At each step, we need to pick a formula on either the left or the right of the turnstile, and 'split' on its topmost connective. Splitting on the connective means using the appropriate left or right rule (depending on whether the chosen formula was to the left or the right of the turnstile), to produce one or more new sequents.

In this case, let's choose the single formula on the right, whose main connective is implication. The implication-right rule is, recall

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\,{\to_{\mathcal{R}}}$$

Setting $\Gamma = A \to (B \to C)$, $\phi = A \wedge B$ and $\psi = C$, we can begin our search for the proof as follows:

$$\frac{A \wedge B, A \to (B \to C) \vdash C}{A \to (B \to C) \vdash (A \wedge B) \to C}\,{\to_{\mathcal{R}}}$$

Since the newly introduced sequent is not in the form of axiom, we must find a formula in it to split, in an attempt to reduce all the leaves of the derivation tree to axioms. Let us split on the left hand formula, $A \to (B \to C)$. Recall that the implication-left rule is

$$\frac{\Gamma \vdash \phi \qquad \Gamma, \psi \vdash \theta}{\Gamma, \phi \to \psi \vdash \theta}\,{\to_{\mathcal{L}}}$$

Setting $\Gamma = A \wedge B$, $\phi = A$, $\psi = B \to C$, and $\theta = C$, we introduce two new sequents

$$\frac{\dfrac{A \wedge B \vdash A \qquad A \wedge B, B \to C \vdash C}{A \wedge B, A \to (B \to C) \vdash C}\,{\to_{\mathcal{L}}}}{A \to (B \to C) \vdash (A \wedge B) \to C}\,{\to_{\mathcal{R}}}$$

This now gives us two leaves on the derivation tree, neither of which are yet axioms. Taking the first of these, there is only one formula with a connective, and conjunction-left applies to give

$$\frac{\dfrac{\dfrac{A \vdash A}{A \wedge B \vdash A}\,{\wedge_{\mathcal{L}}} \qquad A \wedge B, B \to C \vdash C}{A \wedge B, A \to (B \to C) \vdash C}\,{\to_{\mathcal{L}}}}{A \to (B \to C) \vdash (A \wedge B) \to C}\,{\to_{\mathcal{R}}}$$

Since $A \vdash A$ is an axiom, we do not need to split any further on this branch of the tree. We can now split on $A \wedge B$ in the second leaf to give

$$\dfrac{\dfrac{\dfrac{A \vdash A}{A \wedge B \vdash A}\wedge_{\mathcal{L}} \qquad \dfrac{B, B \to C \vdash C}{A \wedge B, B \to C \vdash C}\wedge_{\mathcal{L}}}{A \wedge B, A \to (B \to C) \vdash C}\to_{\mathcal{L}}}{A \to (B \to C) \vdash (A \wedge B) \to C}\to_{\mathcal{R}}$$

Now we split on $B \to C$ using $\to_{\mathcal{L}}$ to give

$$\dfrac{\dfrac{\dfrac{A \vdash A}{A \wedge B \vdash A}\wedge_{\mathcal{L}} \qquad \dfrac{\dfrac{B \vdash B \qquad B, C \vdash C}{B, B \to C \vdash C}\to_{\mathcal{L}}}{A \wedge B, B \to C \vdash C}\wedge_{\mathcal{L}}}{A \wedge B, A \to (B \to C) \vdash C}\to_{\mathcal{L}}}{A \to (B \to C) \vdash (A \wedge B) \to C}\to_{\mathcal{R}}$$

Finally, we have to use weakening on the left to get

$$\dfrac{\dfrac{\dfrac{A \vdash A}{A \wedge B \vdash A}\wedge_{\mathcal{L}} \qquad \dfrac{\dfrac{B \vdash B \qquad \dfrac{C \vdash C}{B, C \vdash C}\mathit{Weakening}_{\mathcal{L}}}{B, B \to C \vdash C}\to_{\mathcal{L}}}{A \wedge B, B \to C \vdash C}\wedge_{\mathcal{L}}}{A \wedge B, A \to (B \to C) \vdash C}\to_{\mathcal{L}}}{A \to (B \to C) \vdash (A \wedge B) \to C}\to_{\mathcal{R}}$$

We have now reached a point where all the leaves on the derivation tree are axioms: we have completed the proof.

Problems with Cut

The bottom up procedure just outlined is non-deterministic in that at most stages one has a choice about which formula to split on. This non-determinism is greatly increased by the presence of the cut rule. The difficulyt with the cut rule is that it can be applied to any formula, and split the sequents up in any way. This is unlike the other rules, that single out formulas with a particular main connective, or at least clearly indicate just one way in which the sequents should be split.

The advantage of cut is that if you are clever, you can find proofs by cutting in known lemmas. The disadvantage is that if you are applying the rules systematically and blindly to search for a proof (e.g. like a computer program), the search may not terminate. It is therefore important to know that, in the case of intuitionistic and classical propositional logic, the cut rule can be eliminated without affecting the range of theorems provable.

2.2.4 Quantifier in Sequent Calculus

For the sake of completeness, we list the left and right rules for the universal and existential quantifiers. The rules for classical logic are:

$$\frac{\Gamma, A[x/t] \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}\forall_{\mathcal{L}} \qquad \frac{\Gamma \vdash A[x/y], \Delta}{\Gamma \vdash \forall x.A, \Delta}\forall_{\mathcal{R}}$$

$$\frac{\Gamma, A[x/y] \vdash \Delta}{\Gamma, \exists x.A \vdash \Delta}\exists_{\mathcal{L}} \qquad \frac{\Gamma \vdash A[x/t], \Delta}{\Gamma \vdash \exists x.A, \Delta}\exists_{\mathcal{R}}$$

(where y must not occur free in Γ or Δ). The rules for intuitionistic logic are just the single conclusion versions of the above.

2.2.5 Sequent Calculus and Natural Deduction

We now describe how to convert a sequent calculus proof into a natural deduction proof for the implication-conjunction fragment of intuitionistic logic. This technique is taken from [GirardLafontTaylor]. Some notes of caution, however. First, the mapping is many-one: several quite distinct sequent proofs can sometimes map onto the same natural deduction proof. Second, the mapping does not hold for all logical systems. In general there is a kind of subsumption ordering amongst proof styles:

- Axiomatic systems cover the widest range of logical systems
- Sequent calculi can be given for only a somewhat narrower range of logics
- Natural deduction systems can only be given for a relatively few logics

Classical and intuitionistic logic are fortunate in having all three kinds of proof system available. Propositional modal logics cover the full range: some like S5 have a natural deduction system; some like K have sequent calculi but no natural deduction; and others only have axiomatic systems.

1. Axioms translate as simple one step ND derivations

   $$A \vdash A \qquad \Longrightarrow \qquad A$$

   We descend down the sequent derivation from its axiom leaves, translating the other steps as follows:

2. Right rules translate into introductions

$$\dfrac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \wedge B}\wedge_{\mathcal{R}} \qquad \Longrightarrow \qquad \dfrac{\begin{matrix}\Gamma & \qquad & \Gamma' \\ \vdots & & \vdots \\ A & & B\end{matrix}}{A \wedge B}\wedge_{\mathcal{I}}$$

$$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\to_{\mathcal{R}} \qquad \Longrightarrow \qquad \dfrac{\begin{matrix}\Gamma, [A]^i \\ \vdots \\ B\end{matrix}}{A \to B}\to_{\mathcal{I},i}$$

Here, the vertical dots indicate ND derivations that have already been constructed using the mapping rules.

3. Left rules translate into eliminations (but written backwards!)

$$\dfrac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\wedge_{\mathcal{L}} \qquad \Longrightarrow \qquad \begin{matrix}\dfrac{\Gamma \qquad \dfrac{A \wedge B}{A}\wedge_{\mathcal{E}}}{\;} \\ \vdots \\ C\end{matrix}$$

$$\dfrac{\Gamma \vdash A \qquad \Gamma', B \vdash C}{\Gamma, \Gamma', A \to B \vdash C}\to_{\mathcal{L}} \qquad \Longrightarrow \qquad \begin{matrix}\dfrac{\Gamma' \qquad \dfrac{\begin{matrix}\Gamma \\ \vdots \\ A\end{matrix} \qquad A \to B}{B}\to_{\mathcal{E}}}{\;} \\ \vdots \\ C\end{matrix}$$

4. The structural rules translate into management of hypotheses.

$$\dfrac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\mathit{Contraction} \qquad\qquad \dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\mathit{Weakening}_{\mathcal{L}}$$

Contraction corresponds to giving two assumptions of A the same index. Weakening corresponds to a null assumption of A

5. Finally, the cut rule corresponds to putting two derivations together.

$$\dfrac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\mathit{cut} \qquad \Longrightarrow \qquad \begin{matrix}\dfrac{\Delta,\ \begin{matrix}\Gamma \\ \vdots \\ A\end{matrix}}{\;} \\ \vdots \\ B\end{matrix}$$

To illustrate the fact that multiple sequent derivations can map onto a single natural deduction derivation, here is a simple example. Both of the sequent derivations

$$\dfrac{\dfrac{\dfrac{A \vdash A \qquad B \vdash B}{A, B \vdash A \wedge B}\wedge_{\mathcal{R}}}{A \wedge A', B \vdash A \wedge B}\wedge_{\mathcal{L}}}{A \wedge A', B \wedge B' \vdash A \wedge B}\wedge_{\mathcal{L}} \qquad\qquad \dfrac{\dfrac{\dfrac{A \vdash A \qquad B \vdash B}{A, B \vdash A \wedge B}\wedge_{\mathcal{R}}}{A, B \wedge B' \vdash A \wedge B}\wedge_{\mathcal{L}}}{A \wedge A', B \wedge B' \vdash A \wedge B}\wedge_{\mathcal{L}}$$

correspond to the same natural deduction derivation

$$\dfrac{\dfrac{A \wedge A'}{A}\wedge_{\mathcal{E}} \qquad \dfrac{B \wedge B'}{B}\wedge_{\mathcal{E}}}{A \wedge B}\wedge_{\mathcal{I}}$$

The two sequent proofs linearize the two parallel $\wedge_{\mathcal{E}}$ natural deduction steps as different orders of applying $\wedge_{\mathcal{L}}$.

In general, the rules of the sequent calculus can be seen as more or less complex combinations of natural deduction rules, and sequent proofs as linearizations of natural deduction proofs.

2.3 Proof Normalization

When are two proofs the same? This section discusses techniques for converting alternate forms of proofs to (minimal) canonical versions of the proof. In natural deduction, this technique is known as proof normalization. In sequent calculus it corresponds to cut elimination. As mentioned in Chapter 2, proof normalization in natural reduction is closely related to lambda-reduction on proof terms produced by the Curry-Howard isomorphism. This is the topic of the section after this.

2.3.1 Normalization in Natural Deduction

β- and η-Reduction

Consider the derivation

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ A\end{matrix} \qquad \begin{matrix}\mathcal{D}_2\\ \vdots\\ B\end{matrix}}{A \wedge B}\wedge_{\mathcal{I}}}{A}\wedge_{\mathcal{E}}$$

Clearly the introduction of the conjunction followed by its immediate elimination is an unnecessary detour in the proof. We could get the same result more simply as

$$\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\end{array}$$

More generally, any step in a derivation that introduces a connective only to immediately eliminate it again at the next step gives rise to a detour. These can be eliminated by the following rules of β-reduction.

Conjunction

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\end{array}\qquad\begin{array}{c}\mathcal{D}_2\\ \vdots\\ B\end{array}}{A\wedge B}\,\wedge_{\mathcal{I}}}{A}\,\wedge_{\mathcal{E}}\qquad\Longrightarrow_{\beta}\qquad\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\end{array}$$

Implication

$$\dfrac{\dfrac{\begin{array}{c}[A]^i\\ \vdots\\ B\end{array}}{A\rightarrow B}\,{\rightarrow_{\mathcal{I},i}}\qquad\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\end{array}}{B}\,{\rightarrow_{\mathcal{E}}}\qquad\Longrightarrow_{\beta}\qquad\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\\ \vdots\\ B\end{array}$$

Disjunction

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\end{array}}{A\vee B}\,\vee_{\mathcal{I}}\qquad\begin{array}{c}[A]^i\\ \vdots\\ C\end{array}\qquad\begin{array}{c}[B]^j\\ \vdots\\ C\end{array}}{C}\,\vee_{\mathcal{E}i,j}\qquad\Longrightarrow_{\beta}\qquad\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A\\ \vdots\\ C\end{array}$$

There are also some lesser known rules of η-reduction that apply when an elimination is immediately by an introduction. For example

Implication

$$\dfrac{\dfrac{[A]^i \qquad \begin{array}{c}\mathcal{D}_1\\ \vdots \\ A \to B\end{array}}{B}\,\to_{\mathcal{E}}}{A \to B}\,\to_{\mathcal{I},i} \qquad \Longrightarrow_{\eta} \qquad \begin{array}{c}\mathcal{D}_1\\ \vdots\\ A \to B\end{array}$$

Conjunction

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \vdots\\ A \wedge B\end{array}}{A}\,\wedge_{\mathcal{I}} \qquad \begin{array}{c}\mathcal{D}_2\\ \vdots\\ B\end{array}}{A \wedge B}\,\wedge_{\mathcal{I}} \qquad \Longrightarrow_{\eta} \qquad \begin{array}{c}\mathcal{D}_1\\ \vdots\\ A \wedge B\end{array}$$

A derivation to which none of these reductions can be applied is said to be in $\beta\eta$-normal form.

Commuting Conversions

Recall the rule for disjunction elimination, with its parasitic formula C that bears no relation to the disjunction $A \vee B$ being eliminated

$$\dfrac{\Gamma \vdash A \vee B \qquad \Gamma, A^i \vdash C \qquad \Gamma, B^j \vdash C}{\Gamma \vdash C}\,\vee_{\mathcal{E}i,j}$$

This parasitic formula can cause problems. Consider the derivation

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ A \vee B\end{array} \qquad \dfrac{\begin{array}{c}[A],[D]\\ \vdots\\ E\end{array}}{D \to E}\,\to_{\mathcal{I}} \qquad \dfrac{\begin{array}{c}[B],[D]\\ \vdots\\ E\end{array}}{D \to E}\,\to_{\mathcal{I}}}{D \to E}\,\vee_{\mathcal{E}} \qquad \begin{array}{c}\mathcal{D}\\ \vdots\\ D\end{array}}{E}\,\to_{\mathcal{E}}$$

This derivation is in β-normal form, but nonetheless it contains a detour. We could get the same result by replacing the assumption of D on the two branches of the disjunction by the actual derivation $\mathcal{D}$ of D, and remove some unnecessary $\to_{\mathcal{I}}$ and $\to_{\mathcal{E}}$ steps:

$$\dfrac{\begin{array}{c}\vdots\\ A \vee B\end{array} \qquad \begin{array}{c}[A],\ \begin{array}{c}\mathcal{D}\\ \vdots\\ D\end{array}\\ \vdots\\ E\end{array} \qquad \begin{array}{c}[B],\ \begin{array}{c}\mathcal{D}\\ \vdots\\ D\end{array}\\ \vdots\\ E\end{array}}{E}\,\vee_{\mathcal{E}}$$

Whenever an elimination follows a parasitic rule, we would like to push (or commute) the elimination upwards one level, so that it stands some chance of being adjacent to an introduction rule that actually mentions the parasitic formula. This gives rise for the following commuting conversions for the two rules introducing parasitic formulas.

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ A\vee B\end{matrix}\quad \begin{matrix}[A]\\ \vdots\\ C\end{matrix}\quad \begin{matrix}[B]\\ \vdots\\ C\end{matrix}}{C}\vee_{\mathcal{E}}\quad \begin{matrix}\mathcal{D}\\ \vdots\end{matrix}}{D}R_{\mathcal{E}} \quad\Longrightarrow_c\quad \dfrac{\begin{matrix}\vdots\\ A\vee B\end{matrix}\quad \dfrac{\begin{matrix}[A]\\ \vdots\\ C\end{matrix}\quad \begin{matrix}\mathcal{D}\\ \vdots\end{matrix}}{D}R_{\mathcal{E}}\quad \dfrac{\begin{matrix}[B]\\ \vdots\\ C\end{matrix}\quad \begin{matrix}\mathcal{D}\\ \vdots\end{matrix}}{D}R_{\mathcal{E}}}{D}\vee_{\mathcal{E}}$$

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ \bot\end{matrix}}{A}\bot_{\mathcal{E}}\quad \begin{matrix}\mathcal{D}_2\\ \vdots\end{matrix}}{B}R_{\mathcal{E}} \quad\Longrightarrow_c\quad \dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ \bot\end{matrix}}{B}\bot_{\mathcal{E}}$$

where $R_{\mathcal{E}}$ stands for any of the elimination rules.

Properties of Normalization

Two important properties of proof normalization in natural deduction are

- Church-Rosser property:
  This says that any proof has a unique normal form.

  Put another way, the order in which the various normalizing conversions are applied to a derivation does not affect the final result.

- Strong Normalization:
  This says that the application of normalizing conversions will eventually terminate to give a normal form proof

2.3.2 Cut-Elimination in Sequent Calculus

Corresponding to normalization in natural deduction is cut elimination in the sequent calculus. The cut rule

$$\dfrac{\Gamma\vdash A \qquad \Delta, A\vdash B}{\Gamma,\Delta\vdash B}\,cut$$

allows us to combine two derivations. In practice the rule is very useful, since it allows us the possibility of proving various lemmas (or useful patterns of inference), and then cutting them into proofs whenever we need them. Cut-elimination says that whenever we cut lemmas in in this way, there is always an equivalent proof that does not use the lemmas, but derives everything from scratch.

Eliminating cuts is a process of moving them up the derivation tree until we cut against an axiom, where it is clear that the cut can be eliminated as follows:

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \vdots\\ \Gamma \vdash A\end{array} \qquad A \vdash A}{A \vdash A}\,\mathit{cut} \qquad \Longrightarrow \qquad \begin{array}{c}\mathcal{D}\\ \vdots\\ \Gamma \vdash A\end{array}$$

We will not go into all the details of how to float cuts up through a derivation. The following examples will give some idea of what is involved.

- Conjunction:
  Replace

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,\wedge_{\mathcal{R}} \qquad \dfrac{\Delta, A \vdash C}{\Delta, A \wedge B \vdash C}\,\wedge_{\mathcal{L}}}{\Gamma, \Delta \vdash C}\,\mathit{cut}$$

by the derivation

$$\dfrac{\Gamma \vdash A \qquad \Delta, A \vdash C}{\Gamma\Delta \vdash C}$$

- Implication:

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,\to_{\mathcal{R}} \qquad \dfrac{\Delta \vdash A \qquad \Delta, B \vdash C}{\Delta, A \to B \vdash C}\,\to_{\mathcal{L}}}{\Gamma, \Delta \vdash C}\,\mathit{cut}$$

is replaced by

$$\dfrac{\dfrac{\Delta \vdash A \qquad \Gamma, A \vdash B}{\Delta, \Gamma \vdash B}\,\mathit{cut} \qquad \dfrac{\Delta, B \vdash C}{\Delta, \Gamma, B \vdash C}\,\mathit{Weakening}_{\mathcal{R}}}{\Gamma, \Delta \vdash C}\,\mathit{cut}$$

- Disjunction

$$\dfrac{\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\,\vee_{\mathcal{R}} \qquad \dfrac{\Delta, A \vdash C \qquad \Delta, B \vdash C}{\Delta, A \vee B \vdash C}\,\vee_{\mathcal{L}}}{\Gamma, \Delta \vdash C}\,\mathit{cut}$$

is replaced by

$$\frac{\Gamma \vdash A \qquad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\,\mathit{cut}$$

Note how pushing the cut upwards has the effect of reducing the number of connectives in the proof. The same is true for the other rules we have not shown for pushing cut upwards. Because of this reduction in complexity, the elimination procedure terminates.

Consequences of Cut Elimination

Gentzen's proof of cut elimination for classical and intuitionistic logic in 1934 was a major step, allowing a number of important logical results to be proved. Amongst them were the consistency and decidability of the sequent calculus. Two other properties of cut-free proofs worth noting are

- Subformula property:
  In a cut free proof of $\Gamma \vdash \phi$, all the formula occuring in the proof are subformulas of either Γ or ϕ.

  By insepection, one can see that all the sequent calculus rules except cut are subformula preserving. The absence of the sub-formula property for cut means that it is a highly non-deterministic rule for use in proof search — the formulas in a sequent give us no clues about whether or not to apply it at any stage.

- Disjunction property:
  If $\vdash A \vee B$, either $\vdash A$ or $\vdash B$

One reminder: although cut elimination can convert sequent proofs to canonical cut-free proofs, the equivalence classes of proofs defined by cut elimination are broader than the equivalence classes defined by normalization in natural deduction. Essentially, alternate linearizations of a normal form ND proof will still count as alternate sequent proofs.

2.4 Curry-Howard Isomorphism

The Curry-Howard Isomorphism (CHI) connects (constructive) logics with type theory and the lambda-calculus. As we saw briefly in Chapter 1, natural deduction rules can be paired with operations in the lambda-calculus. These operations combine terms representing proofs of propositions to build more complex terms representing proofs of propositions.

For example, suppose P represents a proof of the proposition $A \rightarrow B$, and x represents some arbitrary proof of A. That is, P and x are *proof terms* for the two propositions. We pair a proposition with its proof term by means of a colon, e.g.

$$P : A \rightarrow B$$

with the term (P) conventionally written on the left. Here is a simple natural deduction proof showing how more complex proof terms can be constructed:

$$\dfrac{\dfrac{[x : A]^1 \qquad P : A \to B}{P(x) : B}\,\to_{\mathcal{E}}}{\lambda x.P(x) : A \to B}\,\to_{\mathcal{I},1}$$

Here, the rule of $\to_{\mathcal{E}}$ gives rise to *application* of proof terms, and $\to_{\mathcal{I}}$ to lambda abstraction. In this section we will extend this pairing of ND rules with term operators to cover conjunction, disjunction and negation.

Recalling the discussion of η-normalization from the last section, note that the proof is not on β-normal form. The rule for η-reduction says that we can eliminate a detour to give a simpler proof:

$$P : A \to B$$

The proof normalization step of η-reduction is so-called because it really does correspond to doing a η-reduction on the corresponding proof terms

$$\lambda x.P(x) \Longrightarrow_{\eta} P$$

Likewise, the normalization step of β-reduction really does correspond to doing a β-reduction on the corresponding proof terms.

We can think about the CHI under various slogans following slogans

- *Terms as Proofs*
  The expression

  $$\lambda x.P(x) : A \to B$$

  can be read one way as saying that $\lambda x.P(x)$ is a proof of $A \to B$. Given that the term is an abstraction, we can tell that the last step of the proof introduced an implication into the proof $P(x)$. Moreover, this step must have discharged an assumption $x : A$. We can also tell that $P(x)$ must be a proof of B, and that its last step was an implication elimination, applying something whose proof was P to something whose proof was x. Since we know that it was A's proof that was x, we can therefore tell that P was a proof of $A \to B$. Since the term P has no internal structure, we can tell that $A \to B$ was a premise.

  In other words, one can look at the proof term decorating a formula in a natural deduction proof, and reconstruct the proof from the term.

- *Propositions as Types*
  We have just seen how the expression

  $$\lambda x.P(x) : A \to B$$

can be read one way as saying that $\lambda x.P(x)$ is a proof of the proposition $A \to B$. We can also read it as saying that the *type* of the proof $\lambda x.P(x)$ is $A \to B$. In fact, we can view implication, $\to$, as type forming operator familiar from type theory.

It is important to note that there can usually be several proofs of the same type (i.e. several ways of proving the same proposition). For example, consider

$$\dfrac{\dfrac{[x:A]^1 \quad P:A\to B}{P(x):B}\to_{\mathcal{E}}}{\lambda x.P(x):A\to B}\to_{\mathcal{I},1} \qquad P:A\to B \qquad \dfrac{Q:C\to(A\to B)\quad R:C}{Q(R):A\to B}\to_{\mathcal{E}}$$

That is, the proofs $\lambda x.P(x)$, P and $Q(R)$ all have the type $A \to B$. And although $\lambda x.P(x)$ and P are equivalent by η-conversion, and so in some sense represent the same proof, the proof $Q(R)$ is not equivalent to either of these.

We can look on a type / proposition, therefore, as being a set of proofs.

- *Proofs as Programs*
  Here is another example of different proofs of the same type: we let the proposition N also stand for the type Number. We can look at the integers, 1, 2, 3,..., as being different atomic proofs of the fact that there are numbers. We can also look on multiplication and addition as being things which when given two numbers will prove that you can form a third number. Two possible proofs N (that there is at least one number) are

$$\dfrac{\dfrac{\times:N\to(N\to N)\qquad 2:N}{\times(2):N\to N}\qquad \dfrac{\dfrac{+:N\to(N\to N)\qquad 3:N}{+(3):N\to N}\qquad 4:N}{+(3)(4):N}}{\times(2)(+(3)(4)):N}$$

$$\dfrac{\dfrac{+:N\to(N\to N)\qquad 3:N}{\times(3):N\to N}\qquad \dfrac{\dfrac{\times:N\to(N\to N)\qquad 2:N}{\times(2):N\to N}\qquad 4:N}{\times(2)(4):N}}{+(3)(\times(2)(4)):N}$$

Note how very different these proofs are. One constructs the number $2\times(3+4) = 14$ as a proof of N, and the other constructs $3 + (2 \times 4) = 11$.

We can look on these proofs as being two different programs for computing numbers. The correspondence between programs and proofs lies at the heart of functional programming in computer science. It turns out that normalization corresponds to performing computations in a functional language. To give another example, suppose that we do not take all numbers as primitive, but instead start with just

0 and the successor function s that adds one to zero. Thus we could represent the number 3 as the term $s(s(s(0)))$. We can also represent a function for adding 2 to x as a term $\lambda x.s(s(x))$ of type $N \to N$. Now consider a proof

$$\frac{\lambda x.s(s(x)) : N \to N \qquad s(s(s(0))) : N}{\lambda x.s(s(x))[s(s(s(0)))] : N}$$

Lambda reduction of the final proofterm, $\lambda x.s(s(x))[s(s(s(0)))]$ yields $s(s(s(s(s(0)))))$, which is our internal representation for 5, the result of adding 2 to 3.

Note: this discussion of programs as proofs is not required for anything that follows. However, some of the applications of linear logic to computer science have revolved around the design of linear functional programming languages.

2.4.1 CHI for Intuitionistic ND

Let us introduce some operations on proof terms besides application and abstraction. These further operations will be paired with the introduction and elimination rules for other connectives.

Conjunction We have pairing and projection to deal with conjunction. Given two terms, P and Q, we represent the conjunction of them as the ordered pair $\langle P, Q\rangle$. We then have two projection functions fst and snd to get at the first and second elements of an ordered pair. Note that $\mathsf{fst}(\langle P, Q\rangle) = P$ and $\mathsf{snd}(\langle P, Q\rangle) = Q$

We have the following introduction and elimination rules for conjunction

$$\frac{P : A \qquad Q : B}{\langle P, Q\rangle : A \wedge B}\wedge_{\mathcal{I}} \qquad\qquad \frac{M : A \wedge B}{\mathsf{fst}(M) : A}\wedge_{\mathcal{E}1} \quad \frac{M : A \wedge B}{\mathsf{snd}(M) : B}\wedge_{\mathcal{E}2}$$

Notice how we have to be careful about distinguishing two versions of the elimination rules, since one projects onto the first term in the pair, and the other projects onto the second.

Disjunction For disjunction, we want to represent a disjunctive proof of P or Q as a kind of 'atrophied pair', along with an indication of which of P or Q is the one that makes the disjunction true. We can represent this atrophied pair of terms as a single term, plus an indication of whether it is the term for the left or right disjunct. Using inl and inr to signal left and right, we have the introduction rules for disjunction:

$$\frac{P : A}{\mathsf{inl}(P) : A \vee B}\vee_{\mathcal{I}1} \qquad\qquad \frac{P : B}{\mathsf{inr}(P) : A \vee B}\vee_{\mathcal{I}2}$$

Note once again how we have to be careful to distinguish two versions of the introduction rule, depending on whether it is the left or right disjunct that makes the disjunction true.

The term operation corresponding to disjunction elimination is rather more complex. It resembles a "case" statement in programming. The term

$$[\mathsf{case}\, M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)]$$

is to be read as saying that if term M is of the form $\mathsf{inl}(x)$, then the resulting term is P, and if it is of the form $\mathsf{inr}(y)$ the resulting term is Q. The elimination rule is thus

$$\dfrac{M : A \vee B \qquad \begin{array}{c}[x:A]^i\\ \vdots\\ P:C\end{array} \qquad \begin{array}{c}[y:B]^j\\ \vdots\\ Q:C\end{array}}{[\mathsf{case}\, M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] : C}\,\vee\mathcal{E}_{i,j}$$

Note how the x and y in the case construction unify with the terms labelling the assumptions of A and B. This is to ensure that the appropriate proof term for either A or B gets plugged in to either P or Q respectively.

Falsum Rather than treat negation as primitive, we will define it as implication into falsum, i.e.

$$\neg A \; = \; A \rightarrow \bot$$

We then associate an "abort" term with the $\bot_{\mathcal{E}}$ rule

$$\dfrac{M : \bot}{\mathsf{abort}_A : A}\,\bot_{\mathcal{E}}$$

Note how this rule just throws away the proof term M. The 'abort' means that if we prove A by means of proving falsum, we would be well-advised to abort that particular proof.

Term Assignment System

Putting these rules together, we obtain the natural deduction plus term assignment system for propositional intuitionistic logic shown in figure 2.6 We should note that (1) premises are just assigned arbitrary (unique) constants as the proof terms, and (2) proof terms for assumptions should be unique, with the exception of multiple occurrences of the same assumption (i.e. assumptions that are co-indexed).

Proof and Term Reduction

The proof reduction rules described in section 2.3 all lead to corresponding reductions of proof terms. The consequences of β-reduction are shown in figure 2.7 (we use the notation

Introduction

$$\dfrac{\begin{matrix}[x:A]^i\\ \vdots\\ P:B\end{matrix}}{\lambda x.P : A \to B}\to_{\mathcal{I},i}$$

$$\dfrac{P:A \quad Q:B}{\langle P,Q\rangle : A \wedge B}\wedge_{\mathcal{I}}$$

$$\dfrac{P:A}{\mathsf{inl}(P) : A \vee B}\vee_{\mathcal{I}1} \qquad \dfrac{P:B}{\mathsf{inr}(P) : A \vee B}\vee_{\mathcal{I}2}$$

Elimination

$$\dfrac{Q:A \qquad P:A \to B}{P(Q):B}\to_{\mathcal{E}}$$

$$\dfrac{P:A\wedge B}{\mathsf{fst}(P):A}\wedge_{\mathcal{E}1} \qquad \dfrac{P:A\wedge B}{\mathsf{snd}(P):B}\wedge_{\mathcal{E}2}$$

$$\dfrac{M:A\vee B \quad \begin{matrix}[x:A]^i\\ \vdots\\ P:C\end{matrix} \quad \begin{matrix}[y:B]^j\\ \vdots\\ Q:C\end{matrix}}{[\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] : C}\vee_{\mathcal{E}i,j}$$

$$\dfrac{P:\bot}{\mathsf{abort}_A : A}\bot_{\mathcal{E}}$$

Figure 2.6: Term Assignment for Intuitionistic Propositional Logic

$P[M/x]$ to mean term P with all occurrences of x replaced by M). If we look just at the terms, these reductions amount to the following

- $\mathsf{fst}(\langle P,Q\rangle) = P$
  $\mathsf{snd}(\langle P,Q\rangle) = Q$
- $(\lambda x.P)(Q) = P[Q/x]$
- $[\mathsf{case}\ \mathsf{inl}(M)\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] = P[M/x]$
  $[\mathsf{case}\ \mathsf{inr}(M)\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] = Q[M/y]$

The commuting conversions also give rise to term reducitons, including the following

- $[\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)](R)$
  $= [\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ P(R))\ \ (\mathsf{inr}(y)\ Q(R))]$
- $\mathsf{fst}([\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)])$
  $= [\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ \mathsf{fst}(R))\ \ (\mathsf{inr}(y)\ \mathsf{fst}(R))]$
- $\mathsf{snd}([\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)])$
  $= [\mathsf{case}\ M\ \ (\mathsf{inl}(x)\ \mathsf{snd}(R))\ \ (\mathsf{inr}(y)\ \mathsf{snd}(R))]$

Conjunction

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ P : A\end{matrix} \qquad \begin{matrix}\mathcal{D}_2\\ \vdots\\ Q : B\end{matrix}}{\langle P, Q\rangle : A \wedge B}\wedge_{\mathcal{I}}}{\mathsf{fst}(\langle P, Q\rangle) : A}\wedge_{\mathcal{E}} \qquad \Longrightarrow_{\beta} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ P : A\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ P : A\end{matrix} \qquad \begin{matrix}\mathcal{D}_2\\ \vdots\\ Q : B\end{matrix}}{\langle P, Q\rangle : A \wedge B}\wedge_{\mathcal{I}}}{\mathsf{snd}(\langle P, Q\rangle) : B}\wedge_{\mathcal{E}} \qquad \Longrightarrow_{\beta} \qquad \begin{matrix}\mathcal{D}_2\\ \vdots\\ Q : B\end{matrix}$$

Implication

$$\dfrac{\dfrac{\begin{matrix}[x : A]^{i}\\ \vdots\\ P : B\end{matrix}}{\lambda x.P : A \to B}\to_{\mathcal{I},i} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ Q : A\end{matrix}}{(\lambda x.P)(Q) : B}\to_{\mathcal{E}} \qquad \Longrightarrow_{\beta} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ Q : A\\ \vdots\\ P[x/Q] : B\end{matrix}$$

Disjunction

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ M : A\end{matrix}}{\mathsf{inl}(M) : A \vee B}\vee_{\mathcal{I}} \qquad \begin{matrix}[x : A]^{i}\\ \vdots\\ P : C\end{matrix} \qquad \begin{matrix}[y : B]^{j}\\ \vdots\\ Q : C\end{matrix}}{[\mathsf{case}\ \mathsf{inl}(M)\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] : C}\vee_{\mathcal{E}i,j} \qquad \Longrightarrow_{\beta} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ M : A\\ \vdots\\ P[M/x] : C\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ M : B\end{matrix}}{\mathsf{inr}(M) : A \vee B}\vee_{\mathcal{I}} \qquad \begin{matrix}[x : A]^{i}\\ \vdots\\ P : C\end{matrix} \qquad \begin{matrix}[y : B]^{j}\\ \vdots\\ Q : C\end{matrix}}{[\mathsf{case}\ \mathsf{inl}(M)\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] : C}\vee_{\mathcal{E}i,j} \qquad \Longrightarrow_{\beta} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ M : A\\ \vdots\\ Q[M/y] : C\end{matrix}$$

Figure 2.7: Beta Reduction

- $(\mathsf{abort}_A(M))(N) = \mathsf{abort}_B(M)$
  $\mathsf{fst}(\mathsf{abort}_{\langle A,B\rangle(M)}) = \mathsf{abort}_A(M)$
  $\mathsf{snd}(\mathsf{abort}_{\langle A,B\rangle(M)}) = \mathsf{abort}_B(M)$

2.4.2 CHI and Constructivity

The proof terms assigned by the Curry-Howard isomorphism bear a close and non-accidental resemblance to the constructive, proof-based semantics lying behind the Brouwer-Heyting-Kolmogorov (BHK) interpretation for intuitionistic logic (see p. **??**).

We noted that a BHK style interpretation was not readily available for classical logic. For similar reasons, a Curry-Howard isomorphism is not readily available either.[7] The culprit can once again be seen as the classical rule of *reductio ad absurdum*

$$\dfrac{\begin{matrix}[x : A \to \bot] \\ \vdots \\ P : \bot\end{matrix}}{?? : A}\,RAA$$

Bearing in mind the rule $\bot_{\mathcal{E}}$, to derive A we throw away the proof of $\bot$ from $A \to \bot$. If we do not throw this proof away, its abort type will cause the rest of the proof to be aborted. This then leaves the question of what would be a sensible proof term to put in place of "??". There is no obvious answer.

It is thus important to realize that the Curry-Howard Isomorphism does not apply to all logics. As we will see in a moment, it only applies to logics that have a natural deduction formulation. And as we have just seen, it does not even apply to all logics that do have a natural deduction formulation. Nonetheless, there is something of a cottage industry in constructing term assignment systems for various logics, since these can (amongst other things) be useful for designing new forms of functional programming langauge.

2.4.3 Term Assignments for Sequents and Axioms

The section on converting sequent proofs to natural deduction proofs showed that there is not a 1-1 mapping. Distinct sequent proofs can map onto the same natural deduction proof. Because of this, there is no real Curry-Howard Isomorphism for the sequent calculus. One can nevertheless design term assignment systems for the sequent calculus. But there will not be the same close parallels between term reduction and proof normalization as there are for natural deduction.

One term assignment scheme in effect just gives each sequent calculus rule a unique label. This certainly allows one to uncover the structure of a proof from a proof term. But it

[7] The $\lambda\mu$-calculus represents a recent attempt at providing a CHI for classical logic [Parigot]. However, this relies on finding ways of looking at classical logic in a more constructive light.

does nothing to ensure that alternate versions of the same (natural deduction) proof will have terms that can be reduced to a common expression.

Another scheme for term assignment makes use of the mapping from sequent proofs to natural deduction proofs. As a result, it assigns terms that do reduce to common expressions for equivalent sequent proofs. However, this behaviour is inherited from the natural deduction system, and is not inherent to the sequent calculus. The term assignment system obtained in shown in figure 2.8.

It is also possible to give term assignments for axiomatic systems. In the case of propositional logic, this gives rise to combinatory logic.

2.5 Summary

One arrives at linear logic by looking at the proof theory of traditional logics. This is why we have spent so long reviewing the basics of proof theory (such material is not usually covered in introductory logic texts for linguists). So how might we sum matters up to set the scene for the development of linear logic?

First recall that our general aim is to gain access to the actual proof objects underlying the (syntactic) representations of proofs as derivations in systems like natural deduction or sequent calculus. Given that we can only access proofs via their surface forms as derivations, natural deduction looks very promising.

For intuitionistic logic, the Curry-Howard Isomorphism provides us with access to terms representing proofs. Moreover, these terms have non-trivial identity criteria, which allow us (in conjunction with proof normalization) to say when two syntactically distinct derivations correspond to the same proof.

However, natural deduction retains some embarrassing features.

- A Curry-Howard Isomporphism cannot be given for classical logic. Dyed-in-the-wool constructivists might well retort ‘so much the worse for classical logic.’ But for the less ideologically motivated, the failure to find interesting proof objects for classical logic casts doubt on the general aim of elevating the status of proofs in logic.

- The natural deduction formulation of classical logic is not completely symmetrical. Although most of the connectives have their meanings defined by introduction and elimination rules, negation requires three rules: introduction, elimination, and *reductio ad absurdum*. This smears the meaning of negation across a number of rules, that makes a simple characterisation of its proof-theoretic meaning impossible.

- For both classical and intuitionistic logic, the parastitic rules of or-elimination and $\perp$-elimination, are ugly. The commuting conversions these gives rise to considerably complicates the identity criteria of proofs, as defined by proof normalization.

With respect to classical logic, the sequent calculus fares much better. First, the complete symmetry lost in the ND formulation is regained in the sequent formulation of classical

$$\frac{\Gamma \vdash P : B}{\Gamma, x : A \vdash P : B}\ \mathit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{abort}_B : B}\ \mathit{Weakening}_{\mathcal{R}}$$

$$\frac{\Gamma, x : A, y : A \vdash P : B}{\Gamma, z : A \vdash P[z/x, z/y] : B}\ \mathit{Contraction}$$

$$\frac{\Gamma, x : A \vdash P : C}{\Gamma, y : A \wedge B \vdash P[\mathsf{fst}(y)/x] : C}\ \wedge_{\mathcal{L}} \qquad \frac{\Gamma \vdash P : A \qquad \Gamma \vdash Q : B}{\Gamma \vdash \langle P, Q \rangle : A \wedge B}\ \wedge_{\mathcal{R}}$$

$$\frac{\Gamma, x : B \vdash P : C}{\Gamma, y : A \wedge B \vdash P[\mathsf{snd}(y)/x] : C}\ \wedge_{\mathcal{L}}$$

$$\frac{\Gamma, x : A \vdash C \qquad \Gamma, y : B \vdash C}{\Gamma, z : A \vee B \vdash [\mathsf{case}\ z\ \ (\mathsf{inl}(x)\ P)\ \ (\mathsf{inr}(y)\ Q)] : C}\ \vee_{\mathcal{L}} \qquad \frac{\Gamma \vdash P : A}{\Gamma \vdash \mathsf{inl}(P) : A \vee B}\ \vee_{\mathcal{R}}$$

$$\frac{\Gamma \vdash P : B}{\Gamma \vdash \mathsf{inr}(P) : A \vee B}\ \vee_{\mathcal{R}}$$

$$\frac{\Gamma \vdash Q : A \qquad \Gamma, x : B \vdash P : C}{\Gamma, y : A \to B \vdash P[y(M)/x] : C}\ \to_{\mathcal{L}} \qquad \frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x.P : A \to B}\ \to_{\mathcal{R}}$$

$$\frac{\Gamma \vdash Q : A \qquad \Delta, x : A \vdash P : B}{\Gamma, \Delta \vdash P[Q/x] : B}\ \mathit{cut}$$

$$\frac{}{x : A \vdash x : A}\ \mathit{axiom}$$

Figure 2.8: Term Assignment for Sequent Calculus for IPL

logic. In fact it is intuitionistic logic that comes out asymmetric, in that it allows only single conclusion sequents. This is tantamount to dropping the rules of right contraction and weakening, leaving only the left versions. Unfortunately, for both intuitionistic and classical logic the sequent calculus individuates too finely between proofs; derivations that should be identified remain distinct.

Chapter 3

Basic Linear Logic

3.1 Sequent Calculus

The sequent calculus systems for traditional (classical and intuitionistic) logic takes Γ and Δ to be *sets* of formulas in sequents likes $\Gamma \vdash \Delta$. What would happen if instead we took them to be multisets[1] of formulas?

3.1.1 Contraction and Weakening

The first consequence of shifting to multisets would be that the structural rules of contraction and weakening no longer hold. We would need to drop the following rules

$$\frac{\Gamma,\ A,\ A\ \vdash\ B}{\Gamma,\ A\ \vdash\ B}\ \textit{Contraction} \qquad\qquad \frac{\Gamma\ \vdash\ B}{\Gamma,\ A\ \vdash\ B}\ \textit{Weakening}$$

The rule of exchange

$$\frac{\Gamma,\ A,\ B\ \vdash\ C}{\Gamma,\ B,\ A\ \vdash\ C}\ \textit{Exchange}$$

applies equally to sets and to multisets, however.

What is the intuitive significance of contraction and weakening? Weakening opens the door for fake or irrelevant dependencies. Given a derivation of $\Gamma \vdash A$, we can weaken it to $\Gamma, B \vdash A$. This muddies the waters as to whether A really depends on B or not — it is only by looking at the preceding derivation that we can tell. The sequent alone does not say. In the same vein, we saw how in Chapter **??** weakening corresponded to discharging a null assumption in natural deduction. Relevance logic [???] abandoned weakening (while

[1] Multisets are unordered lists of elements, where a count is kept of the number of times each element occurs.

preserving contraction) in order to try and block derivations of "irrelevant" conclusions like

$$A \vdash B \to A$$

which gives the impression that there is some dependency of A on B, when in fact there is no dependency at all.

Contraction allows us to use formulas without counting how many times we use them. Remembering that sequent rules should be read 'upwards', contraction says that if we want to prove $\Gamma, A \vdash B$, we can do so by copying A and proving $\Gamma, A, A \vdash B$. Girard describes contraction as the "fingernail of infinity". He illustrates this with the use of an axiom like $\forall x.\mathsf{integer} x \to \exists y.\mathsf{integer}(y) \wedge y > x$, which says for any integer x there is larger integer y. It is *repeated* use of this axiom, via contraction, that means it furnishes us with an infinite supply of integers. As another aside, without contraction first order-predicate calculus would be decidable.

3.1.2 Multiplicatives and Additives

Additive and Multiplicative Conjunction

The loss of certain structural rules is not the only consequence of moving to multisets, however. Consider the traditional $\wedge_{\mathcal{R}}$ rule, for example

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_{\mathcal{R}}$$

Note how the upper Γ contexts are identical. Suppose instead we were to allow distinct upper contexts Γ and Δ

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \wedge_{\mathcal{R}}{}'$$

where the original rule is just the special case $\Delta = \Gamma$. How would replacing the $\wedge_{\mathcal{R}}$ rule by $\wedge_{\mathcal{R}}{}'$ change the system for traditional logic (where Γ and Δ are sets of formulas)?

The answer is that it changes nothing. The rules with identical and disjoint upper contexts are inter-derivable so long as we can use contraction and weakening:

- $\wedge_{\mathcal{R}}$ from $\wedge_{\mathcal{R}}{}'$

$$\cfrac{\cfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma, \Gamma \vdash A \wedge B} \wedge_{\mathcal{R}}{}'}{\Gamma \vdash A \wedge B} \textit{Contraction}$$

- $\wedge_{\mathcal{R}}{}'$ from $\wedge_{\mathcal{R}}$

$$\dfrac{\dfrac{\Gamma \vdash A}{\Gamma, \Delta \vdash A}\,\textit{Weakening} \qquad \dfrac{\Gamma \vdash B}{\Gamma, \Delta \vdash B}\,\textit{Weakening}}{\Gamma, \Delta \vdash A \wedge B \wedge}$$

However, if we dispense with contraction and weakening, the two rules are no longer interderivable. In fact, the two rules correspond to two different forms of conjunction, $\otimes$ and &:

$$\dfrac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,\otimes_{\mathcal{R}} \qquad\qquad \dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\,\&_{\mathcal{R}}$$

We can play a similar trick with the left rule for conjunction. The following two rules are equivalent given contraction and weakening

$$\dfrac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\,\wedge_{\mathcal{L}} \qquad\qquad \dfrac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\,{\wedge_{\mathcal{L}}}'$$

We can see these are equivalent as follows

- $\wedge_{\mathcal{L}}$ from ${\wedge_{\mathcal{L}}}'$

$$\dfrac{\dfrac{\Gamma, A \vdash C}{\Gamma, A, B \vdash C}\,\textit{Weakening}}{\Gamma, A \wedge B, \vdash C}\,{\wedge_{\mathcal{L}}}'$$

- ${\wedge_{\mathcal{L}}}'$ from $\wedge_{\mathcal{L}}$

$$\dfrac{\dfrac{\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B, B \vdash C}\,\wedge_{\mathcal{L}}}{\Gamma, A \wedge B, A \wedge B \vdash C}\,\wedge_{\mathcal{L}}}{\Gamma, A \wedge B, \vdash C}\,\textit{Contraction}$$

The question arises as to which rule rule should become the left rule for which of the two newly introduced connectives. It turns out that if we make the wrong pairing, we can re-derive contraction and weakening, which would make the distinction between the connectives collapse. For example, consider combining the rule $\&_{\mathcal{R}}$ with the incorrect left rule:

$$\dfrac{\dfrac{A \vdash A \qquad A \vdash A}{A \vdash A \& A}\,\&_{\mathcal{R}} \qquad \dfrac{\Gamma, A, A \vdash B}{\Gamma, A \not\& A \vdash B}\,\not\&_{\mathcal{L}}}{\Gamma, A \vdash B}\,\textit{cut}$$

That is, from $\Gamma, A, A \vdash B$ we can obtain the contracted version $\Gamma, A \vdash B$.

If we make the correct pairings of left and right rules, we get the following two left rules for the new forms of conjunction

$$\frac{\Gamma, A \vdash C}{\Gamma, A\&B \vdash C}\,\&_{\mathcal{L}} \qquad\qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,\otimes_{\mathcal{L}}$$

Tensor $\otimes$ is a *multiplicative* conjunction: its (right) rule combines multiple contexts Γ and With & is an *additive* conjunction: it deals with a single context Γ. Troelstra sometimes calls the multiplicative connectives "context free" — they apply independent of differences in context — and the additive connectives "context sharing" — they only apply if contexts are shared.

Given the left and right rules for the two conjunctions, we can give them an informal explanation as follows

- Multiplicative conjunction $\otimes$
  This allows you to combine premises / resources in a proof into one bundle. The left rule says that if you have two premises A and B you can bundle them together to form a single premise, $A \otimes B$. The right rule says if you have two proofs, one of A from Γ and one of B from Δ, you can combine the proofs and bundle together the results, $A \otimes B$

- Additive conjunction &
  This allows alternatives from a given set of premises to be combined. The right rule says that if there is a proof of A from Γ, and also a proof of B from Γ, these two alternatives can be conjoined to say that there is a proof of $A\&B$ from Γ. This is to be read as saying you have a choice of proving A and of proving B, but not both (since proving A consumes premises that need to be used in proving B and vice versa. The left rule says that if you can prove C from Γ and A, you can also prove it from Γ and $A\&B$. The $A\&B$ again says that you have the choice of using A and of using B to make the derivation work, though in fact you have to choose A.

  Despite being a conjunction & behaves in many ways more like a (meta-level) disjunction. $A\&B$ indicates a free (or internal) choice: we can decide which of A or B to use. Sometimes only one of the alternatives will give us what we need, but we are at least free to choose what works. This is similar to classical disjunction: given A we can weaken this to $A \vee B$; but if we then want to single out one of the disjuncts, the only reliable one to choose is A. This internal or free choice differs from the external or forced choice offered by $\oplus$ (the multiplicative disjunction). $A \oplus B$ says that A and B are alternatives, but we have no power over which one is selected; the choice is made externally.

Additive and Multiplicative Disjunction

Moving to multisets also allows us to distinguish between multiplicative and additive versions of disjunction. Additive implication, plus $\oplus$ has the following left and right rules

$$\frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\,\oplus_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,\oplus_{\mathcal{R}}$$

We can read this as follows

- Additive disjunction $\oplus$
The left rule says that if both Γ and A, and Γ and B prove C, then Γ and some random choice of either A or B will prove C. This random, or external choice of A or B is safe, as we know that they both lead to the same conclusion; so it doesn't matter which one is picked for us. The right rule says that if Γ proves A, then Γ proves A or B. Like the traditional rule for disjunction, this step means that we lose track of whether the disjunction is proved by virtue of either A or B being proved. It hides this information. Any further inferences from $A \oplus B$ had thus better be truly insensitive to which of A or B is the 'real' disjunct.

The multiplicative disjunction $\parr$ is *not* a connective of intuitionistic linear logic, since its right rule only makes sense with multiple conclusion sequents.[2] Recall from chapter **??** that intuitionistic logic requires single conclusion sequents. Multiplicative disjunction has the following left and right rules

$$\frac{\Gamma_1, A \vdash \Delta_1 \qquad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \parr B \vdash \Delta_1, \Delta_2}\,\parr_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\,\parr_{\mathcal{R}}$$

As with conjunction, in the presence of contration and weakening the distinction between the two connectives collapses. Weakening makes the $\parr_{\mathcal{L}}$ rule derivable from *plusL*, and contraction allows the reverse derivation. Weakening (on the right) makes $\oplus_{\mathcal{R}}$ derivable frin $\parr_{\mathcal{R}}$, and contraction on the right allows the reverse derivation. We can try to understand multiplicative disjunction as follows

- Multiplicative Disjunction $\parr$
The right rule says that if Γ proves either A or B, it proves $A \parr B$. That is, commas on the right hand sequents are implicit multiplicative disjunctions, in just the way that commas on the left of sequents are implicit multiplicative conjunctions. If $A \parr B$ holds, then if it is not the case that A holds we can be sure that B holds, and vice versa.

3.1.3 Negation

Negation is not stricly speaking a connective in linear logic. All atomic formulas come in a positive form, A, and a negative form, $A^\perp$. When negation is applied a complex formula, a series of equivalences enable one to push the negation inwards until it only applies to atomic formulas. These equivalences include

$$\begin{array}{lcl@{\qquad\qquad}lcl}
(A \otimes B)^\perp & = & A^\perp \parr B^\perp & (A \parr B)^\perp & = & A^\perp \otimes B^\perp \\
(A \& B)^\perp & = & A^\perp \oplus B^\perp & (A \oplus B)^\perp & = & A^\perp \& B^\perp \\
(!A)^\perp & = & ?A^\perp & (?A)^\perp & = & !A^\perp
\end{array}$$
$$A^{\perp\perp} = A$$

[2] Though there are multiple conclusion formulations of intuitionistic linear logic that do admit $\parr$ as a connective [Paiva].

(see below for explanation of the exponentials ! and ?). A formula like $(A \otimes B^\perp)^\perp$ (where A and B are atomic) is just an alternative notation for $A^\perp \parr B$.

The rules for negation allow one to move formulas to move across the turnstile, flipping their polarities as they go:

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^\perp \vdash \Delta}\ \perp_{\mathcal{L}} \qquad\qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^\perp, \Delta}\ \perp_{\mathcal{R}}$$

Note that when applied to the identity axiom $A \vdash A$, $\perp_{\mathcal{R}}$ yields the sequent

$$\vdash A^\perp, A$$

This can be read as saying that out of nothing one can create a paired consumer and producer of A, which like matter and anti-matter annihilate one another when they meet.

Perhaps a more useful metaphor is to think in terms of *action* and *reaction.* Conservation of 'momentum' ensures that to every action A there is an equal and opposite reaction, $A^\perp$. We can also think of an action of type A as being an answer / output / consumer, and a reaction of type $A^\perp$ as being a question / input / producer. In a sequent $\Gamma \vdash \Delta$, formulas on the left (in Γ) are inputs, and formulas on the right (in Δ) outputs. This is borne out if we use the negation rules to move all of Γ to the right: each formula A in Γ gets negated to $A^\perp$.

In fact, Girard often uses one-sided sequents in presenting linear logic. The one-sided sequent (where if $\Gamma = A_1, \ldots, A_n$, then $\Gamma^\perp = A_1^\perp, \ldots, A_n^\perp$)

$$\vdash \Gamma^\perp, \Delta$$

is equivalent to the two sided sequent

$$\Gamma \vdash \Delta$$

3.1.4 The Exponentials

Linear logic drops the structual rules of weakening and contraction. However a logic without these rules is very weak. Linear logic allows for a controlled way of re-introducing contraction and weakening on specific formulas by means of the exponentials (or modalities) ! and ?. The "of course" exponential, !, allows contraction and weakening on the left hand side of sequents. The "why not" exponential, ?, allows contraction and weakening on the right of sequents.

We can read $!A$ as meaning roughly that the resource A can be duplicated (reproduced) as often or as little as we like. Its dual, $?A$, means roughly that A can be consumed as often or as little as we like.

Another way of thinking about the exponentials is as follows. Atomic propositions in linear logic are a little like signals on a wire: they are created (as input) and immediately consumed (as output). If we take a transient signal like A, then $!A$ corresponds to *storing* the signal in some form of memory. $?A$ corresponds to *reading* from memory.

The sequent rules for the exponentials are (where if $\Gamma = A_1, \ldots, A_n$, $!\Gamma =!A_1, \ldots, !A_n$, and likewise for $?\Gamma$)

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\mathit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\,\mathit{Weakening}_{\mathcal{R}}$$

$$\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\mathit{Contraction}_{\mathcal{L}} \qquad \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\,\mathit{Contraction}_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\mathit{Dereliction}_{\mathcal{L}} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\,\mathit{Dereliction}_{\mathcal{R}}$$

$$\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\,\mathit{Promotion}_{\mathcal{L}} \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\,\mathit{Promotion}_{\mathcal{R}}$$

Promotion corresponds to storing things in memory. Read *Promotion*$_{\mathcal{R}}$ as saying that if duplicable inputs $!\Gamma$ give rise to a transient output A (or the repeatedly consumable $?\Delta$), we can duplicate Γ some more to as many copies of A as we want. So it is safe to store A in memory. We can rearrange *Promotion*$_{\mathcal{L}}$ (using the negation rules) to

$$\frac{!\Gamma \vdash A^{\perp}, ?\Delta}{!\Gamma, \vdash !(A^{\perp}), ?\Delta}\,\mathit{Promotion}_{\mathcal{L}}$$

which says that if duplicable $!\Gamma$ gives a transient input $A^{\perp}$, we can duplicate this input as much as we want; i.e. it can be stored.

Weakening corresponds to erasing from memory. If Δ follows from Γ, it continues to follows from Γ plus whatever else we choose to ignore / erase from memory. Contraction corresponds to copying from memory. Dereliction corresponds to reading from memory.

The exponentials are sometimes known as modals. This is because the rules defining them are similar in form to the modal $\Box$ and $\Diamond$ operators of S4 modal logic. Also not that there inter-derivability under negation

$$!A = (?A^{\perp})^{\perp}$$

is similar to

$$\Box A = \neg\Diamond(\neg A)$$

3.1.5 Implication

(Multiplicative) implication can be defined classically as

$$A \multimap B =_{df} A^{\perp} \parr B$$

(c.f. the traditional classical definition $A \to B =_{df} \neg A \vee B$). However, par is only a classical connective: it requires multiple conclusion sequents. Therefore, this definition will not do for intuitionistic versions of linear logic. Fortunately, we can take implication as primitive, and employ the rules

$$\frac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\multimap_{\mathcal{L}} \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\multimap_{\mathcal{R}}$$

It is also possible to define an additive form of implication, but we will not go into this — see [Troelstra:lnll], ch4. It turns out not to be very interesting.

3.1.6 The Identities

In traditional classical logic, the constants verum and falsum ($\top$ and $\bot$) are identities, in much the same way that 1 and 0 are in arithmetic. The identities map conjunction and disjunction onto themselves

- $A \wedge \top \equiv A$ (cf $N \times 1 = N$)
- $A \vee \bot \equiv A$ (cf $N + 0 = N$)

But now we have two forms of conjunction and disjunction, so we need two forms of identity.

- Multiplicative identities: $\mathbf{1}$ and $\bot$:

  $(A \otimes \mathbf{1}) \equiv A$
  $(A \parr \bot) \equiv A$

  $\mathbf{1}^{\bot} = \bot \qquad \bot^{\bot} = \mathbf{1}$

- Additive identities: $\top$ and $\mathbf{0}$:

  $(A \& \top) \equiv A$
  $(A \oplus \mathbf{0}) \equiv A$

  $\top^{\bot} = \mathbf{0} \qquad \mathbf{0}^{\bot} = \top$

The rules for the identities are as follows

$$\frac{}{\Gamma, \mathbf{0} \vdash \Delta}\mathbf{0}_{\mathcal{L}} \qquad\qquad \frac{}{\Gamma \vdash \top, \Delta}\top_{\mathcal{R}}$$

$$\frac{}{\bot \vdash}\bot_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\bot_{\mathcal{R}}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\mathbf{1}_{\mathcal{L}} \qquad\qquad \frac{\vdash \mathbf{1}}{}\mathbf{1}_{\mathcal{R}}$$

Intuitively, we can think of the identities as follows

- Unit: **1**
  The trivial resource that can be produced from nothing. Another way of putting this is that if a collection of resources produces **1** (and nothing else), then we can consume / throw away that collection of resources

- Top: $\top$
  Top consumes all resources

- Imposibility: **0**
  The impossible resource, or something that will produce any resource (in the same way that $\top$ consumes all resources).

- Bottom: $\perp$
  This is the resource that can be consumed by nothing. It represents resources (premises) left over and unused in a derivation.

3.1.7 Intuitionistic and Classical Linear Logic

As with traditional logic, intuitionistic linear logic is obtained by restricting ourselves to single conclusion sequents. This is tantamount to ruling out any possibility of contraction and weakening on the left. Classical linear logic employs multiple conclusion sequents. Figures 3.1 and 3.2 show the sequent systems for intuitionistic and classical linear logic.

Note that not all the connectives and constants used in classical linear logic are available for intuitionistic linear logic.

Note also that the cut rule can be eliminated for both classical and intuitionistic linear logic.

3.2 Constructivity in Linear Logic

Both classical and intuitionistic linear logic are constructive. This is unlike traditional logic, where the move to multiple conclusion sequents destroys constructivity. So why is it that a multiple conclusion sequent calculus for linear logic can still be constructive?

The brief answer is that it is uncontrolled contraction and weakening on the right that loses constructivity. In the traditional case, single conclusion sequents rule out at a stroke the possibility of contraction and weakening on the right. This leaves us with intuitionistic logic, which allows uncontrolled contraction and weakening on the left. But linear logic does not allow uncontrolled contraction and weakening on either the left or the right. It is therefore unnecessary to control the application of these rules on the right by the sweeping means of single conclusion sequents. We have the necessary degree of control to ensure constructivity, even with classical multiple conclusion sequents.

$$\frac{}{A \vdash A}\,\mathit{axiom} \qquad \frac{\Gamma \vdash A \qquad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,\mathit{cut}$$

$$\frac{\Gamma \vdash A \qquad \Delta, B \vdash c}{\Gamma, \Delta, A \multimap B \vdash C}\,{\multimap_{\mathcal{L}}} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,{\multimap_{\mathcal{R}}}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,\otimes_{\mathcal{L}} \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,\otimes_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash C}{\Gamma, A\&B \vdash C}\,\&_{\mathcal{L}1} \quad \frac{\Gamma, B \vdash C}{\Gamma, A\&B \vdash C}\,\&_{\mathcal{L}2} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A\&B}\,\&_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\,\oplus_{\mathcal{L}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,\oplus_{\mathcal{R}1} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\,\oplus_{\mathcal{R}2}$$

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B}\,\mathit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\,\mathit{Contraction}_{\mathcal{L}}$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}\,\mathit{Dereliction}_{\mathcal{L}} \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\mathit{Promotion}_{\mathcal{L}}$$

$$\frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A}\,\mathbf{1}_{\mathcal{L}} \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1}_{\mathcal{R}}$$

$$\frac{}{\Gamma, \mathbf{0} \vdash A}\,\mathbf{0}_{\mathcal{L}} \qquad \frac{}{\Gamma \vdash \top}\,\top_{\mathcal{R}}$$

Figure 3.1: Sequent Calculus: Intuitionistic Linear Logic

$$\frac{}{A \vdash A}\,\textit{axiom} \qquad \frac{\Gamma_1 \vdash A, \Delta_1 \qquad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\textit{cut}$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \multimap B \vdash \Delta_1, \Delta_2}\,\multimap_{\mathcal{L}} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\,\multimap_{\mathcal{R}}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\,\otimes_{\mathcal{L}} \qquad \frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2, \vdash A \otimes B, \Delta_1, \Delta_2}\,\otimes_{\mathcal{R}}$$

$$\frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \parr B \vdash \Delta_1, \Delta_2}\,\parr_{\mathcal{L}} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\,\parr_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \& B \vdash \Delta}\,\&_{\mathcal{L}1} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \& B \vdash \Delta}\,\&_{\mathcal{L}2} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \& B, \Delta}\,\&_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\,\oplus_{\mathcal{L}} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\,\oplus_{\mathcal{R}1} \quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\,\oplus_{\mathcal{R}2}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\textit{Weakening}_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\,\textit{Weakening}_{\mathcal{R}}$$

$$\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\textit{Contraction}_{\mathcal{L}} \qquad \frac{\Gamma, \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\,\textit{Contraction}_{\mathcal{R}}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,\textit{Dereliction}_{\mathcal{L}} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\,\textit{Dereliction}_{\mathcal{R}}$$

$$\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\,\textit{Promotion}_{\mathcal{L}} \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\,\textit{Promotion}_{\mathcal{R}}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\,\mathbf{1}_{\mathcal{L}} \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1}_{\mathcal{R}}$$

$$\frac{}{\Gamma, \mathbf{0} \vdash \Delta}\,\mathbf{0}_{\mathcal{L}} \qquad \frac{}{\Gamma \vdash \top, \Delta}\,\top_{\mathcal{R}}$$

$$\frac{}{\bot \vdash}\,\bot_{\mathcal{L}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, \vdash \bot, \Delta}\,\bot_{\mathcal{R}}$$

Figure 3.2: Sequent Calculus: Classical Linear Logic

Of course, this answer raises a further question. Why does contraction and weakening on the right lead to non-constructivity? The remainder of this section attempts to answer this question, following the discussion in [GirLafTay:pat].

Keeping Track of Proofs

The discussion of the non-constructive nature of traditional classical logic pointed the finger of blame at the natural deduction rule of *reductio ad absurdum*:

$$\dfrac{\begin{array}{c}\neg A\\ \vdots\\ \bot\end{array}}{A}\,RAA$$

The criticism was that any proof resulting in $\bot$ needs to be 'thrown away' in case it infects anything that depends with it. The rule of RAA not only throws away the proof of $\bot$, it also introduces a new, and unnconnected proof of A. There is no way of keeping track of where exactly this new proof came from.

In the sequent calculus formulation of classical logic, the notion of (not) 'keeping track' of where proofs came from takes a somewhat different form. There is nothing like the rule of RAA to blame. Instead, in a sequent system contraction and weakening are the culprits.

As mentioned at the beginning of this chapter, weakening (on the left) allows one to introduce fake dependencies: given that $\Gamma \vdash A$ one can weaken this to $\Gamma, B \vdash A$, where it is not immediately obvious that B contributes nothing to the derivation of A. Weakening on the right allows you to go from $\Gamma \vdash A$ to $\Gamma \vdash A, B$ where it no longer obvious which of A or B follows from Γ.

Suppose we have two proofs of B

$$\begin{array}{c}\mathcal{D}_1\\ \vdots\\ \vdash B\end{array} \qquad\qquad \begin{array}{c}\mathcal{D}_2\\ \vdots\\ \vdash B\end{array}$$

By means of contraction and weakening on the right, plus cut, we can combine these two proofs as follows

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \vdots\\ \vdash B\end{array}}{\vdash C, B}\,Weakening_{\mathcal{R}} \qquad \dfrac{\begin{array}{c}\mathcal{D}_2\\ \vdots\\ \vdash B\end{array}}{C \vdash B}\,Weakening_{\mathcal{L}}}{\vdash B, B}\,cut}{\vdash B}\,Contraction_{\mathcal{R}}$$

To eliminate the cut by pushing it upwards, we have a choice of whether to view the cut as being on the B or the C formula in $\vdash B, C$. Unfortunately, the choice leads to two very different cut-free proofs, namely:

$$\begin{array}{ccc} \mathcal{D}_1 & \qquad & \mathcal{D}_2 \\ \vdots & & \vdots \\ \vdash B & & \vdash B \end{array}$$

What this means is that there is a single proof that can reduce, via cut-elimination to either $\mathcal{D}_1$ or $\mathcal{D}_2$. Which is to say, however different these proofs are, classical logic regards them as being the same proof. Or put another way, classical logic says that all proofs of a particular type (e.g. B, $A \to B$, etc) are identical. This is sometimes expressed has saying that classical logic has no interesting denotational semantics (beyond the normal truth-conditional semantics that identifies all proofs of the same proposition).

To sum up more informally. Contraction and weakening on the left make it hard to tell which premises were used to derive a particular conclusion. But contraction and weakening on the right are worse: they make it hard to tell which conclusion came from the premises.

Classical linear logic is constructive, i.e. it allows us to keep track of where proofs come from, because it limits the application of contraction and weakening. Traditional intuitionistic logic is constructive because it completely prohibits contraction and weakening on the right.

3.3 Natural Deduction

Up until now, we have considered linear logic as a development of sequent calculus. Sequent calculus can be seen as a kind of meta-level description of how natural deduction proofs are carried out, which includes (extraneous) information about the order in which certain inference steps were carried out. One might therefore expect a smooth transition to natural deduction formulations of linear logic. Such formulations can be given, but for Girard at any rate they are less satisfactory than the sequent formulations.

3.3.1 Tensor Fragment

If we confine our attention to the fragment of linear logic comprising only implication and tensor, plus !, we can formulate the natural deduction system shown in figure 3.3.

Discharging Assumptions The restrictions on discharging assumptions are stricter for the linear system than the non-linear systems we have seen previously: exactly one occurrence of an assumption must be discharged. Discharging zero occurrences corresponds to weakening, and more than one to contraction.

Parasitic Rules The elimination rule for tensor is parasitic in the same way that the disjnuction elimination rule was for traditional natural deduction. This leads to additional commuting conversions when it comes to the normalization of linear natural deduction proofs.

$$\dfrac{\begin{matrix}[A]^i\\ \vdots\\ B\end{matrix}}{A \multimap B}\multimap_{\mathcal{I},i} \qquad \dfrac{A \qquad A \multimap B}{B}\multimap_{\mathcal{E}}$$

$$\dfrac{A \qquad B}{A \otimes B}\otimes_{\mathcal{I}} \qquad \dfrac{A \otimes B \qquad \begin{matrix}[A]^i[B]^j\\ \vdots\\ C\end{matrix}}{C}\otimes_{\mathcal{E},i,j}$$

$$\dfrac{}{\mathbf{1}}\mathbf{1}_{\mathcal{I}} \qquad \dfrac{A \qquad \mathbf{1}}{A}\mathbf{1}_{\mathcal{E}}$$

$$\dfrac{!A \qquad B}{B}\mathit{Weakening} \qquad \dfrac{!A \qquad \begin{matrix}[!A]^i[!A]^j\\ \vdots\\ C\end{matrix}}{C}\mathit{Contraction}_{i,j}$$

$$\dfrac{!A}{A}\mathit{Dereliction} \qquad \dfrac{!A_1 \ldots !A_n \qquad \begin{matrix}[!A_1^i \ldots !A_n^k]\\ \vdots\\ B\end{matrix}}{!B}\mathit{Promotion}_{i \ldots k}$$

Figure 3.3: Natural Deduction for Tensor-Implication Fragment

Exponentials The promotion rule (or !-introduction) is not the version given in [Troelstra:lnll], but the corrected one of [BBPH].

3.3.2 Gentzen Style ND

The additive connectives need to share the same contexts (i.e. premises and assumptions.) There is no natural way of representing this in the Prawitz style natural deduction scheme used for the implication tensor fragment. Instead, a sequent style natural deduction system make the sharing or otherwise of contexts explicit. The resulting system for intuitionistic linear logic is shown in figure 3.4

To obtain natural deduction for classical linear logic, we take the rules for intuitionistic linear logic, plus

- Define negations as
$$A^{\perp} =_{df} A \multimap \perp$$
(Note that $\perp$ is the unit for $⅋$, so that $A^{\perp} ⅋ \perp \equiv A^{\perp}$. If we define $A \multimap B =_{df} A^{\perp} ⅋ B$, we can then see that $A \multimap \perp \equiv A^{\perp}$.)
- Add the rule
$$\frac{\Gamma, A \multimap \perp \vdash \perp}{\Gamma \vdash A}$$
- Define
$A ⅋ B =_{df} (A^{\perp} \otimes B^{\perp})^{\perp}$
$?A =_{df} (!A^{\perp})^{\perp}$

3.3.3 Normalization and Term Assignment

Proof normalization and the Curry-Howard isomorphism carry across from the natural deduction system for traditional intuitionistic logic to linear intuitionistic logic.[3] As before, normalization corresponds to cut-elimination in the sequent system. One could thus inherit a term-assignment system for sequent formulations of linear logic from the natural deduction system. However, proof nets (next chapter) provide a more interesting and direct way assigning proof terms for sequent systems.

Normalization

We will not list all the reduction rules for linear natural deduction (in particular, reductions for the additive connectives are essentially the same as for non-linear conjunction and disjunction). For example, we have

[3] Though dealing with the exponentials is not entirely straightforward.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap_{\mathcal{I}} \qquad \frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \multimap_{\mathcal{E}}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes_{\mathcal{I}} \qquad \frac{\Gamma \vdash A \otimes B \qquad \Delta A, B \vdash C}{\Gamma, \Delta \vdash C} \otimes_{\mathcal{E}}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B} \&_{\mathcal{I}} \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash A} \&_{\mathcal{E}1} \frac{\Gamma \vdash A \& B}{\Gamma \vdash B} \&_{\mathcal{E}2}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \oplus_{\mathcal{I}1} \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B} \oplus_{\mathcal{I}2} \qquad \frac{\Gamma \vdash A \oplus B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Gamma, \Delta \vdash C} \oplus_{\mathcal{E}}$$

$$\frac{}{\vdash \mathbf{1}} \mathbf{1}_{\mathcal{I}} \qquad \frac{\Gamma \vdash \mathbf{1} \qquad \Delta \vdash A}{\Gamma, \Delta \vdash A} \mathbf{1}_{\mathcal{E}}$$

$$\frac{}{\Gamma \vdash \top} \top_{\mathcal{I}} \qquad \frac{}{\Gamma, \mathbf{0} \vdash A} \mathbf{0}_{\mathcal{E}}$$

$$\frac{\Gamma \vdash !A \qquad \Delta, !A, !A \vdash B}{\Gamma, \Delta \vdash B} \mathit{Contraction}(!_{\mathcal{E},c})$$

$$\frac{\Gamma_1 \vdash !A_1, \ldots, \Gamma_n \vdash !A_n \qquad !A_1, \ldots, !A_n \vdash B}{\Gamma_1, \ldots, \Gamma_n \vdash !B} !_{\mathcal{I}} \qquad \frac{\Gamma \vdash !B}{\Gamma \vdash B} \mathit{Dereliction}(!_{\mathcal{E},d})$$

$$\frac{\Gamma \vdash !A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash B} \mathit{Weakening}(!_{\mathcal{E},w})$$

$$\frac{}{A \vdash A} \mathit{axiom}$$

Figure 3.4: Sequent ND for Intuitionistic Linear Logic

$$\dfrac{\dfrac{\begin{matrix}A\vdash A\\ \vdots\\ \Gamma,A\vdash B\end{matrix}}{\Gamma\vdash A\multimap B}{\scriptstyle\multimap_{\mathcal{I}}} \qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ \Delta\vdash A\end{matrix}}{\Gamma,\Delta\vdash B}{\scriptstyle\multimap_{\mathcal{E}}} \qquad\Longrightarrow_\beta\qquad \begin{matrix}\mathcal{D}_1\\ \vdots\\ \Delta\vdash A\\ \vdots\\ \Gamma,\Delta\vdash B\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots\\ \Gamma_1\vdash A\end{matrix}\quad\begin{matrix}\mathcal{D}_2\\ \vdots\\ \Gamma_2\vdash B\end{matrix}}{\Gamma_1,\Gamma_2\vdash A\otimes B}{\scriptstyle\otimes_{\mathcal{I}}} \qquad \begin{matrix}A\vdash A\quad B\vdash B\\ \vdots\\ \Delta,A,B\vdash C\end{matrix}}{\Gamma_1,\Gamma_2,\Delta\vdash C}{\scriptstyle\otimes_{\mathcal{E}}} \qquad\Longrightarrow_\beta\qquad \begin{matrix}\mathcal{D}_1\qquad\mathcal{D}_2\\ \vdots\qquad\quad\vdots\\ \Gamma_1\vdash A\quad\Gamma_2\vdash B\\ \vdots\\ \Gamma_1,\Gamma_2,\Delta\vdash C\end{matrix}$$

$$\dfrac{\dfrac{\Gamma_1\vdash !A_1,\ldots,\Gamma_n\vdash !A_n \qquad \begin{matrix}!A_1\vdash !A_1,\ldots,!A_n\vdash !A_n\\ \vdots\\ !A_1,\ldots,!A_n\vdash B\end{matrix}}{\Gamma_1,\ldots,\Gamma_n\vdash !B}\text{Prom}}{\Gamma_1,\ldots,\Gamma_n\vdash B}\textit{Dereliction} \qquad\Longrightarrow_\beta\qquad \begin{matrix}\Gamma_1\vdash !A_1,\ldots,\Gamma_n\vdash !A_n\\ \vdots\\ \Gamma_1,\ldots,\Gamma_n\vdash B\end{matrix}$$

(There are also reductions for a promotion followed by a contraction, and a promotion followed by a weakening.) Commuting conversions are also required for the parasitic rules like $\otimes_{\mathcal{E}}$, e.g.

$$\dfrac{\dfrac{A\otimes B \qquad \begin{matrix}[A][B]\\ \vdots\\ C\end{matrix}}{C}{\scriptstyle\otimes_{\mathcal{E}}} \qquad \begin{matrix}\mathcal{D}\\ \vdots\end{matrix}}{D}{\scriptstyle r} \qquad\Longrightarrow_c\qquad \dfrac{A\otimes B \qquad \dfrac{\begin{matrix}[A][B]\\ \vdots\\ C\end{matrix}\qquad\begin{matrix}\mathcal{D}\\ \vdots\end{matrix}}{D}{\scriptstyle r}}{D}{\scriptstyle\otimes_{\mathcal{E}}}$$

Term Assignment

Figure 3.5 shows the term assignment for the intuitionistic linear logic. The assignment introduces some new term constructors. The additive connectives make use of the constructors familiar from intuitionistic logic.

let **and** $\otimes$ Introduction of $A\otimes B$ gives rise to a tensor pairing of proof terms, $a\times b$, where a and b are the proof terms of A and B. However, this is not an ordinary pairing: we are not allowed to project down onto the individual elements a and b. This is unlike

$$\frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x.f : A \multimap B} \multimap_{\mathcal{I}} \qquad \frac{\Gamma \vdash f : A \multimap B \qquad \Delta \vdash a : A}{\Gamma, \Delta \vdash f(a) : B} \multimap_{\mathcal{E}}$$

$$\frac{\Gamma \vdash a : A \qquad \Delta \vdash b : B}{\Gamma, \Delta \vdash a \times b : A \otimes B} \otimes_{\mathcal{I}} \qquad \frac{\Gamma \vdash a : A \otimes B \qquad \Delta x : A, y : B \vdash f : C}{\Gamma, \Delta \vdash \mathsf{let}\ a\ \mathsf{be}\ x \times y\ \mathsf{in}\ f : C} \otimes_{\mathcal{E}}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \& B} \&_{\mathcal{I}} \qquad \frac{\Gamma \vdash p : A \& B}{\Gamma \vdash \mathsf{fst}(p) : A} \&_{\mathcal{E}1} \quad \frac{\Gamma \vdash p : A \& B}{\Gamma \vdash \mathsf{snd}(p) : B} \&_{\mathcal{E}2}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}(a) : A \oplus B} \oplus_{\mathcal{I}1} \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathsf{inr}(b) : A \oplus B} \oplus_{\mathcal{I}2}$$

$$\frac{\Gamma \vdash m : A \oplus B \qquad \Delta, x : A \vdash p : C \qquad \Delta, y : B \vdash q : C}{\Gamma, \Delta \vdash [\mathsf{case}\ m\ \ (\mathsf{inl}(x)\ p)\ \ (\mathsf{inr}(y)\ q)] : C} \oplus_{\mathcal{E}}$$

$$\frac{}{\vdash * : \mathbf{1}} \mathbf{1}_{\mathcal{I}} \qquad \frac{\Gamma \vdash i : \mathbf{1} \qquad \Delta \vdash f : A}{\Gamma, \Delta \vdash \mathsf{let}\ i\ \mathsf{be}\ * inf : A} \mathbf{1}_{\mathcal{E}}$$

$$\frac{}{\Gamma \vdash \mathbf{t} : \top} \top_{\mathcal{I}} \qquad \frac{}{\Gamma, x : \mathbf{0} \vdash \mathsf{abort}_x : A} \mathbf{0}_{\mathcal{E}}$$

$$\frac{\Gamma_1 \vdash a_1 :!A_1, \ldots, \Gamma_n \vdash a_n :!A_n \qquad !x_1 : A_1, \ldots, x_n :!A_n \vdash f : B}{\Gamma_1, \ldots, \Gamma_n \vdash \mathsf{promote}\ a_1 \ldots a_n\ \mathsf{for}\ x_1 \ldots x\ \mathsf{in}\ f :!B} \textit{Promotion}$$

$$\frac{\Gamma \vdash a :!A \qquad \Delta \vdash f : B}{\Gamma, \Delta \vdash \mathsf{discard}\ a\ \mathsf{in}\ f : B} \textit{Weakening} \qquad \frac{\Gamma \vdash a :!A}{\Gamma \vdash \mathsf{derelict}(a) : A} \textit{Dereliction}$$

$$\frac{\Gamma \vdash a :!A \qquad \Delta, x :!A, y :!A \vdash f : B}{\Gamma, \Delta \vdash \mathsf{copy}\ a\ \mathsf{as}\ x, y \in f : B} \textit{Contraction}(!_{\mathcal{E},c})$$

$$\frac{}{x : A \vdash x : A} \textit{axiom}$$

Figure 3.5: Term Assignment for Intuitionistic Linear Logic

the additive pairing $\langle a, b\rangle$, where $\mathsf{fst}(\langle a, b\rangle) = a$ and $\mathsf{snd}(\langle a, b\rangle) = b$. Instead, elimination makes use of a let constructor that can do pairwise substitution of elements in tensor pairs

$$\mathsf{let}\ a \times b\ \mathsf{be}\ u \times v\ \mathsf{in}\ f \implies_\beta f[a/u, b/v]$$

Linear Abstraction In the absence of weakening and contraction, all lambda terms are linear. That is, in the expression $\lambda x.f$, the lambda will bind exactly one occurrence of x. The following is a linear lambda term: $\lambda x.p(x)$. The following two are not $\lambda x.y$, $\lambda x.p(x, x)$.

Identities Two constants, $*$ and $\mathbf{t}$ are introduced as proof terms for the identities $\mathbf{1}$ and $\top$.

Exponentials The term constructors for the exponential ! are intended to have mnemonic names. Contraction gives rise to copying variables, weakening to discarding them. A derelicition of a variable corresponds to taking it out of storage, and promotion moves stored variables from one place to another.

Terms for Classical Linear Logic

Since classical linear logic is constructive, one can construct term assignments for it, as well as for intuitionistic linear logic (see [Bierman,Abramsky]). However, the existence of proof nets has in part deflected effort from this enterprise.

3.4 Quantifiers in Linear Logic

The universal and existential quantifiers in linear logic are given the same proof rules as in traditional logic.[4] For the sequent calculus:

$$\frac{\Gamma, A[x/t] \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}\,\forall_{\mathcal{L}} \qquad \frac{\Gamma \vdash A[x/y], \Delta}{\Gamma \vdash \forall x.A, \Delta}\,\forall_{\mathcal{R}}$$

$$\frac{\Gamma, A[x/y] \vdash \Delta}{\Gamma, \exists x.A \vdash \Delta}\,\exists_{\mathcal{L}} \qquad \frac{\Gamma \vdash A[x/t], \Delta}{\Gamma \vdash \exists x.A, \Delta}\,\exists_{\mathcal{R}}$$

(where y must not occur free in Γ or Δ).

For natural deduction:

[4] There has been some work on more intrinsically linear quantifiers, but we will not go into this here.

$$\frac{\Gamma \vdash A[x/y]}{\Gamma \vdash \forall x.A}\,\forall_{\mathcal{I}} \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[x/t]}\,\forall_{\mathcal{E}}$$

$$\frac{\Gamma \vdash A[x/t]}{\Gamma \vdash \exists x.A}\,\exists_{\mathcal{I}} \qquad \frac{\Gamma \vdash \exists x.A \qquad \Delta, A[x/y] \vdash C}{\Gamma, \Delta \vdash C}\,\exists_{\mathcal{E}}$$

(where t free for x, y free for x and not free in A).

However, even though the rules of inference are the familiar ones, the meaning of the quantifiers is subtly changed in the absence of contraction. A formula like $\forall x.\phi(x)$ is not to be read as saying that *all* xs are ϕ. In order for this to hold, we would need to use contraction to repeatedly copy the universal statement and apply it to every x. Instead the formula is to be read as saying that *any* one x will be ϕ, but only the one.

The difference between "any" and "all" is brought out by thinking of what it means when you tell a child they can "pick any cake they like." This normally means that they are at liberty to choose any one cake, not that that they can pick all the cakes.

3.5 Axiomatic System

For the sake of reference, the axiomatic (or Hilbert style) presentation of linear logic is shown in figure 3.6.

3.6 Encoding Traditional Logic

Since the exponentials allow controlled reintroduction of contraction and weakening, it is possible to represent traditional intuitionistic and classical logic inside linear logic.

3.6.1 Intuitionistic Logic

Given a formula A of intuitionistic propositional, its encoding in linear logic, A^i is defined as follows

$$\begin{array}{rcl} A^i & = & A \quad (\text{atomic } A) \\ (A \wedge B)^i & = & A^i \& B^i \\ (A \vee B)^i & = & A^i \oplus B^i \\ (A \rightarrow B)^i & = & !A^i \multimap B^i \\ (\neg A)^i & = & !A^i \multimap \mathbf{0} \\ \perp^i & = & \mathbf{0} \\ A_1, \ldots, A_n \vdash B & = & !A^i_1, \ldots, \, !A^i_n \vdash B^i \end{array}$$

Axioms

1. $A \multimap A$
2. $(A \multimap B) \multimap ((B \multimap C) \multimap (A \multimap C))$
3. $(A \multimap (B \multimap C)) \multimap (B \multimap (A \multimap C))$
4. $A \multimap (B \multimap A \otimes B)$
5. $(A \multimap (B \multimap C)) \multimap ((A \otimes B) \multimap C)$
6. $\mathbf{1}$
7. $\mathbf{1} \multimap (A \multimap A)$
8. $(A \& B) \multimap A, \qquad (A \& B) \multimap B$
9. $((A \multimap B) \& (B \multimap C)) \multimap (A \multimap (B \& C))$
10. $A \multimap (A \oplus B), \qquad B \multimap (A \oplus B)$
11. $((A \multimap C) \oplus (B \multimap C)) \multimap ((A \oplus B) \multimap C)$
12. $A \multimap \top$
13. $\mathbf{0} \multimap A$
14. $B \multimap (!A \multimap B)$
15. $(!A \multimap (!A \multimap B)) \multimap (!A \multimap !B)$
16. $!(A \multimap B) \multimap (!A \multimap !B)$
17. $!A \multimap A$
18. $!A \multimap !!A$

CLL. $((A \multimap \bot) \multimap \bot) \multimap A$

Rules

$\multimap$ $A, A \multimap B \Rightarrow B$

$\otimes$ $A, B \Rightarrow A \otimes B$

! $A \Rightarrow !A$

Comments

Axiom CLL for classical linear logic only

$A^\bot =_{df} A \multimap \bot$

Other connectives defined via negation

Figure 3.6: Hilbert System for Linear Logic

3.6.2 Classical Logic

3.7 Semantics / Applications

For people who are familiar with standard set-based model theory (e.g. Tarskian truth-value semantics for classical propositional and predicate logic, Kripke semantics for modal and intuitionistic logics, etc), there is little in the way of intuitively comprehensible formal semantics for linear logic. This is not to say that there is no semantics for linear logic; on the contrary, there are numerous proposals on the market. But they require some mathematical sophistication to be appreciated.

In this section, we briefly sketch some of the approaches to the semantics of linear logic. The aim is not technical, but to try and harden intuitions about what linear logic means. Semantics can be valuable in finding applications for a logic, and so in some cases we will merely give an informal semantics, indicating how linear logic may be applied in a particular area.

3.7.1 Internal-External Semantics

Perhaps the most intuitive semantics proposed for linear logic is due to [Mitchell]. It is founded on the metaphor that proofs are consumers. Proofs make use of a global resource, shared between an internal and an external environment. The internal environment represents the global resource that has not so far been used / allocated. The external environment represents the global resource that has bee used. A conservation principle demands that the global resource remains the same size, regardless of how it is spread across the internal and external environments.

We use a pair (m, n) to represent the global resource, $m + n$, spread across the internal environment m and the external environment n. An initial distribution is one where none of the global resource has been allocated to the external environment, i.e. $(m + n, 0)$.

An initial forcing relation $\models_0$ is a relation between internal-external pairs and atomic propositions. Essentially $(m, n) \models_0 A$ means that satifying the atomic proposition A requires amount n of resource to be consumed. We recursively define a satisfiability relation $\models$ as follows

- Atomic formulas
  $(m, n) \models A$ iff $(m, n) \models_0 A$

- Tensor $\otimes$
  $(m, n) \models \phi \otimes \psi$ if $\exists\, n_1, n_2$ s.t. $n = n_1 + n_2$:
  $(m + n_2, n_1) \models \phi$ and $(m + n_1, n_2) \models \psi$

  That is, the resource n allocated in satisfying $\phi \otimes \psi$ can be split into two parts. One part, n_1 is needed to satisfy ϕ, and the other n_2 to satisfy ψ. Combining these two parts, $n_1 + n_2 = n$ is enough to satisfy $\phi \otimes \psi$.

- Par $\parr$
  $(m, n) \models \phi \parr \psi$ if $\forall\, m_1, m_2$ s.t. $m = m_1 + m_2$:
  $(m_1, n + m_2) \models \phi$ or $(m_2, n + m_1) \models \psi$

  No matter how much of the internal environment is allocated to the external environment (i.e. consumed) it will be enought to satisfy at least one of either ϕ or ψ. The resource already allocated, n, represents a lower bound on the amount of resource needed to satisfy either ϕ or ψ.

- Negation
  $(m, n) \models \phi^\perp$ if $(n, m) \not\models \phi$

  Your remaining unallocated resource is insufficient to satisfy ϕ

- Implication $\multimap$ $(m, n) \models \phi \multimap \psi$ if $\forall\, m_1, m_2$ s.t. $m = m_1 + m_2$:
  $(n + m_2, m_1) \not\models \phi$ or $(m_2, n + m_1) \models \psi$

  Implication is equivalent to $\phi^\perp \parr \psi$. If $(m, n) \models \phi \multimap \psi$, for any allocation of the remaining internal resource, either (1) it it (m_1) is insufficient (on its own) to satisfy ϕ, or (2) when it (m_2) is combined with the currently allocated resource n it is sufficient to satisfy ψ.

- With &
  $(m, n) \models \phi \& \psi$ if
  $(m, n) \models \phi$ and $(m, n) \models \psi$

  The allocated resource is enough to satisfy ϕ and also enough to satisfy ψ, but not both at once

- Plus $\oplus$
  $(m, n) \models \phi \oplus \psi$ if
  $(m, n) \models \phi$ or $(m, n) \models \psi$

  The allocated resource satisfies one of ϕ or ψ, but we do not specify which.

- Unit **1**
  $(m, n) \models \mathbf{1}$ if $n = 0$

  Unit is satisfied without the consumption of any resources

- Bottom $\perp$
  $(m, n) \models \perp$ if $m \neq 0$

  Bottom is satisfied whenever you still have some unallocated resource. This represents a kind of failure, in that derivations are expected to consume/allocate all of the global resource. However, it is not a devasting form of failure like impossibility (**0**), from which anything follows.

- Top $\top$
  $(m, n) \models \top$ always

Top is always satisfied: it can soak up any unallocated resources. This additive identity is most similar to *verum*/true in traditional logic

- Impossibility $\mathbf{0}$
  $(m,n) \not\models \mathbf{0}$

  Impossibility is never satisfied: c.f. *falsum* in traditional logic

- Of course !
  $(m,n) \models !\phi$ if $n = 0$ and $(m,n) \models \phi$

  This is equivalent to defining $!\phi =_{df} (\mathbf{1} \& \phi)$. That is, ϕ can be satisfied without consuming any resources.

- Why not ?
  $(m,n) \models ?\phi$ if $m \neq 0$ or $(m,n) \models \phi$

  This is equivalent to defining $?\phi =_{df} (\perp \oplus \phi)$. Either ϕ is satisfied, or there is some remaining unallocated resource.

To illustrate this semantics, Mitchell discusses the (infamous) example of buying cigarettes. The formula $\$1 \multimap C$ says that if I spend a dollar (\$1), I can obtain a packet of cigarettes (C). Let $(m,n) \models_0 \$k$ if $n = k$ (i.e. if I have spent k dollars, and thus moved from the internal to the external environment. And for goods G, let $(m,n) \models_0 G$ if the cost of G is $\$n$. It is then possible to show that

$$(m,0) \models \$1 \multimap C \text{ if } (m-1,1) \models C$$

That is, transfering one dollar to the external environment in exchange for the cigarettes C. It can also be shown that

$$(m,0) \models (\$1 \otimes \$1) \multimap (C \otimes C \text{ if } (m-2,2) \models C \otimes C$$

Mitchell goes on to point out that the semantics sketched above is incomplete for linear logic: the model makes $A\&(B \oplus C)$ equivalent to $(A \oplus B)\&(A \oplus C)$, which is not a theorem of linear logic. This arises because the additives are defined as classical $\wedge$ and $\vee$. In order to correct this, Mitchell offers a refinement of the model using quantales. We will not go into this here. Hopefully the incomplete semantics is enought to shed some intuitive light on the linear logic connectives.

3.7.2 Semantics of State

3.7.3 Coherence Spaces

3.8 Fragments of Linear Logic

Table from [Lincoln]

Fragment	Complextity
⊗ ⅋ & ⊕ ! ?	Undecidable
⊗ ⅋ & ⊕	PSPACE complete
⊗ ⅋ ! ?	Unknown (check!)
⊗ ⅋	NP complete

Note that classical propositional logic is NP-complete

Chapter 4

Proof Nets

Proof nets are a new way of representing sequent calculus proofs. Recall that sequent proofs can be seen as linearizations, or sequentializations, of natural deduction proofs. This linearization means that irrelevant differences in the order in which certain proof rules are applied can lead to spurious distinctions being drawn between sequent proofs. For example, on p. **??** we saw how two distinct, cut-free sequent proofs mapped onto the same natural deduction proof. The two sequent proofs only differed in the relative order in which two proof steps were applied. These sequential differences were 'parallelized' out in the natural deduction proof.

For those familiar with formal language theory, the following analogy might help. One can define various forms of derivation for context free grammars: leftmost derivations (where the leftmost non-terminal is always expanded); rightmost derivations (where the rightmost non-terminal is always expanded); top-down derivations; bottom-up derivations; etc. All these different (sequential) derivation regimes give rise to different orders of grammar rule application. Parse trees abstract away from these inessential ordering differences to get to the underlying structure that all these different derivation schemes assign to a string of words. They essentially parallelize those rule applications whose relative orders are immaterial. Thus one might draw a rough parallel between (a) sequent proofs and (rightmost / leftmost / top-down / bottom-up) derivations, and (b) natural deduction proofs and parse trees.

Proof nets provide an alternative means of parallelizing inessential differences in sequent proofs. Given that natural deduction proofs in some ways already do this, one might wonder why an alternative method is required. The answer lies in certain flaws in natural deduction that were pointed out at the end of Chapter 3:

- For (traditional) classical logic, the symmetric pairing of introduction and elimination rules for connectives is lost, thanks to the *reductio ad absurdum rule*. By contrast, the sequent formulation of classical logic is beautifully symmetric.
- Rules like disjunction elimination (and tensor-elimination in linear logic) introduce parasitic formulas in natural deduction formulations. These necessitate the introduc-

tion of quite complex commuting conversions in order to make proof normalization work. Sequent formulations do not introduce parasitic formulas.

- The rules discharging assumptions in natural deduction have a global nature. They need to refer to the whole structure of the foregoing proof to locate the assumption being discharged. In sequent calculus, this information is held locally within the sequent.

This suggests that sequent calculus would be a much better way of representing proofs than natural deduction, were it not for the fact that it introduces inessential distinctions between proofs.[1]

Proof nets have been developed primarily for linear logic, and even then, only for fragments of the logic — the additive connectives pose various problems. There is a hope that proof net methods can be applied more generally to other logics (e.g. traditional intuitionistic logic). But in this chapter, we will focus only on the most basic cases for linear logic.

4.1 Towards Proof Nets

4.1.1 Two Examples

There is a sense in which a linear logic derivation is just an attempt to match up consumers and producers of atomic (propositional) resources. Suppose for example that we wished to prove

$$A,\ A \multimap B \vdash B$$

Note that the negation rule allows us to move each of the premises on the left of the turnstile to the right, negating them as we go. Thus we get

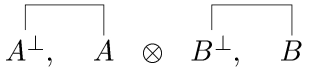
$$\vdash A^{\perp},\ (A \multimap B)^{\perp},\ B$$

Pushing negations inwards, this is the same as

$$\vdash A^{\perp},\ A \otimes B^{\perp},\ B$$

We can connect the positive and negative atomic literals up as follows

(3)

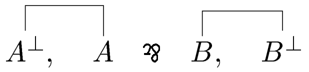


[1] The view that sequent calculus is 'better' is not universally shared. It certainly motivates the development of proof nets. But there are plenty of researchers in linear logic who are not actively engaged with proof nets, and who prefer natural deduction formulation. Against the sequent calculus, one can for example note that it is asymmetric for intuitionistic logic in a way that natural deduction is not. If symmetry is an important property, the argument cuts both ways.

The A and the $A^\perp$ connect, and the B and the $B^\perp$ connect, cancelling one another out. This indicates that we have successfully matched producers, or inputs, ($A^\perp$, $B^\perp$), with consumers or outputs (A, B).

We should straight away point out that not all ways of connecting positive and negative atoms result in the successful matching of consumers and producers. Consider for example the invalid sequent $A, B \not\vdash A \parr B$. Turning this into a one-sided sequent, pushing negation in, and connecting literals gives rise to

(4)



This is *not* a valid way of connecting up positive and negative atoms.

Both (3) and (4) are *proof structures*. However, only (3) is a *proof net*; i.e. a proof structure corresponding to a valid derivation.

The main difference between the two proof structures is in the connective, $\otimes$ or $\parr$. One would, after all, expect conjunction and disjunction to connect up in different ways. Crudely put, conjunction ($\otimes$) brings together the results of two different derivations. Disjunction ($\parr$) characterises the result of a single derivation. The conjunctive structure is OK, as the the A and $B^\perp$ atoms connect up to different (one step, axiomatic) derivations. The disjunctive structure fails, for exactly the same reasons.

The differences between $\otimes$and $\parr$ become clearer if we consider the one-sided sequent formulation of linear logic.

4.1.2 One-Sided Sequent Calculus

The definition of $A \multimap B$ as $A^\perp \parr B$ in classical linear logic, and the ability of negation to move all formulas to the right of sequents permits a very compact formulation of classical linear logic, using one-sided sequents. This is shown in figure 4.1. Consider the rule for par:

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr$$

Note how (i) the disjunction comes from just one context, Γ, and (ii) Γ appear above and below the line. We might thus rewrite the rule as

$$\frac{\begin{array}{ccc} & \Gamma & \\ \vdots & & \vdots \\ A & & B \end{array}}{A \parr B}$$

which both makes the surplus context Γ implicit, but also marks the fact that the context is common to both disjuncts.

Now consider the rule for tensor

$$\frac{}{\vdash A^{\perp}, A}\,\mathit{axiom} \qquad \frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\Gamma, \Delta}\,\mathit{cut}$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\,\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,\parr$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\,\& \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\,\oplus_1 \quad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\,\oplus_2$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, \perp} \qquad \frac{}{\mathbf{1}} \qquad \frac{}{\Gamma, \top}$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,\mathit{Weakening} \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,\mathit{Contraction}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\,\mathit{Dereliction} \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,\mathit{Promotion}$$

Figure 4.1: One-Sided Sequents for Classical Linear Logic

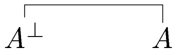
$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes$$

Here the surplus contexts Γ and Δ appear both above and below the line, but note that they are disjoint. Surpressing the surplus contexts, we might rewrite the rule as

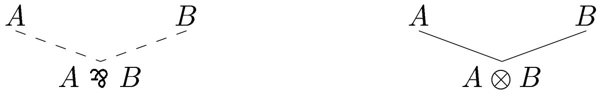
$$\frac{\begin{array}{cc}\Gamma & \Delta\\ \vdots & \vdots\\ A & B\end{array}}{A \otimes B}$$

Here the Γ and Δ express disjointness conditions on how the literals in A and B.

We can also rewrite the axiom rule as



Perhaps a better way of expressing jointness/disjointness conditions on contexts, while surpressing reference to them altogether, is to draw trees with 'hard' and 'soft' links:



The soft links, represented by dashed lines, indicate shared contexts. The hard links, represented by solid lines, indicate disjoint contexts The axiom rule stays as it is (i.e. with a hard link). The notion of hard and soft links makes it easier to understand the notion of a switch graph.

4.1.3 Switch Graphs

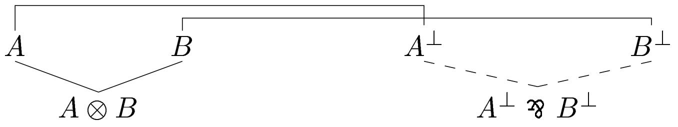
Soft (or par, $\parr$) links point to shared derivations. Therefore, if we break one of the soft links from a par, the other link should still be able to get us to the derivation. On the other hand, hard links point to disjoint derivations. If we follow one link from a tensor, there should not be a cycle (other than through an unbroken soft link) that gets us back to the other link of the tensor.

This idea is formalized using switch graphs.

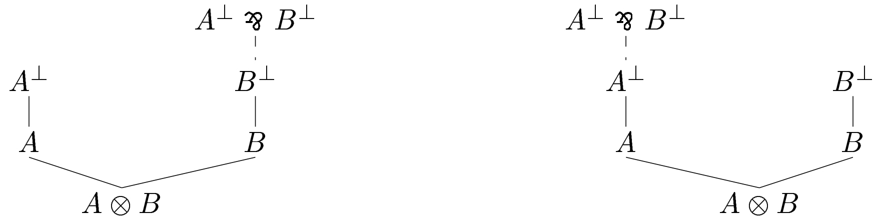
- Given a proof structure Π, a switch graph associated with Π is any sub-graph obtained from Π by taking each par node and deleting one of its soft links while leaving the other one intact.
- A proof structure Π is a proof net iff every switch graph of Π is a tree.

Deleting one of every pair of soft par links should break all the cycles in the proof structure that arise from pars stemming from the same derivation. The remaining tensor links should point to separate derivations, and so should not introduce any cycles Let us illustrate this with some examples.

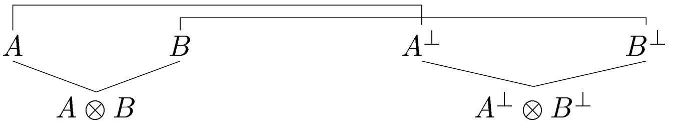
Consider the valid sequent $A \otimes B \vdash A \otimes B$. As a one sided sequent, this becomes $\vdash (A \otimes B)^{\perp}, A \otimes B$. Pushing negations inwards, this becomes $\vdash A^{\perp} \parr B^{\perp}, A \otimes B$. Connecting atoms we get the proof structure



This structure has just one par node. We can break either one of its soft links, and get a tree:



Now consider an invalid sequent $A \otimes B \vdash A \parr B$. This gives rise to the proof structure



This structure does not contains any soft links to be broken. It is also not a tree, as it has a cycle — A; $A^{\perp}$; $A^{\perp} \otimes B^{\perp}$; $B^{\perp}$; B; $A \otimes B$; A. It is therefore not a proof net.

4.1.4 Constructing Proof Structures

Let us informally describe how to prove a sequent using proof nets. We will take as an example the slightly more complex sequent

$$A \otimes B, C \vdash (A \otimes B) \otimes C$$

We proceed as follows

- Move all the formulas on the left of the sequent to the right, negating them as the cross the turnstile. This gives a one-sided sequent.

  For our example, this gives

  $$\vdash (A \otimes B)^{\perp}, C^{\perp}, (A \otimes B) \otimes C$$

- Move negation inwards according to the following identities (note that implications get converted to pars):

$(A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp}$
$(A \parr B)^{\perp} = A^{\perp} \parr B^{\perp}$
$A^{\perp\perp} = A$
$A \multimap B = A^{\perp} \parr B$
$(A \multimap B)^{\perp} = A \parr B^{\perp}$

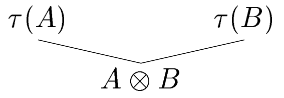
This gives a one-sided sequent in neagtion normal form.

For our example, this gives

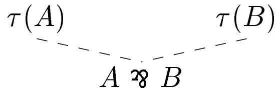
$$\vdash (A^{\perp} \parr B^{\perp}), C^{\perp}, (A \otimes B) \otimes C$$

- Recursively convert the individual formula into trees according to the following rules

  1. $\tau(A) = A, \quad \tau(A^{\perp}) = A^{\perp}$
     for atomic A
  2. $\tau(A \otimes B) =$

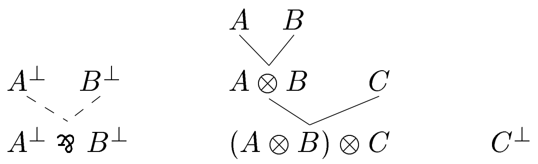


  3. $\tau(A \parr B) =$



  This gives a *proof frame.*

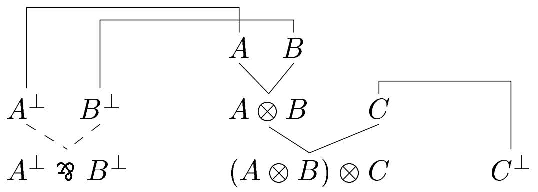
  In our example, this gives rise to the proof frame:



- Connect up positive and negative instances of the same atomic literal. This gives a *proof structure.*

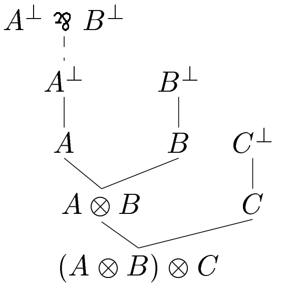
  These atoms will always occur at the leaves of formula trees.
  If there are unbalanced numbers of positive and negative atoms, the sequent is invalid. (But a balanced number does not imply validity).
  If there is more than one positive and negative pair for a given atom, there will be more than one way of connecting atoms. Not all connections, if any, will lead to valid proof nets.

  In our example, we have only one way of linking atoms, and so only one proof structure:



- Determine if at least one of the proof structures obtained is a proof net. Efficient ways of doing this will be discussed in the next section.

  In our example, note that both switch graphs obtainable from the proof structure are trees (acyclic). Thus we have a proof net.

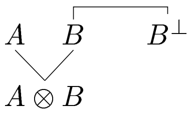


4.1.5 From Sequent Proofs to Proof Nets

We now indicate how to turn a (one-sided) sequent proof into a proof net. For this, we need to define premise (entry) and conclusion (exit) nodes for proof structures. Exploiting the vertical direction in which we have been drawing proof structures on the page, we can do this informally as follows.

- Any node that has no arc entering it from above is a *premise (entry) node*
- Any node that has no arc leaving it from below is an *conclusin (exit) node*

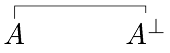
For example, in the (partial) proof structure



the A node is an entry node (it has no arcs coming in to the top of it), and $A \otimes B$ and $B^\perp$ are exit nodes (they have no arcs leaving from beneath them).

The rules for converting a one-sided sequent proof Π to a proof net $\mathcal{N}(\Pi)$ are as follows, and operate recursively

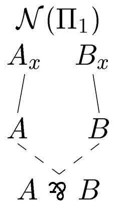
1. If Π is an axiom, $\vdash A, A^\perp$, then $\mathcal{N}(\Pi)$ is the net comprising the single axiom link



2. If Π is of the form

$$\dfrac{\begin{matrix}\Pi_1 \\ \vdots \\ \vdash \Gamma, A, B\end{matrix}}{\vdash \Gamma, A \parr B}\ \parr$$

(where Π_1 is the derivation leading to this last step in Π), then $\mathcal{N}(\Pi)$ is



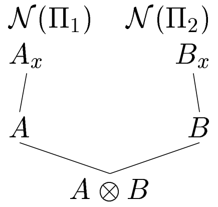
where A_x and B_x are the A and B exit nodes of the proof net for the sub-derivation Π_1

(This means that the entry nodes of $\mathcal{N}(\Pi)$ are the entry nodes of $\mathcal{N}(\Pi_1)$; its exist nodes are those of $\mathcal{N}(\Pi_1)$ minus A_x and B_x, plus $A \otimes B$.)

3. If Π is of the form

$$\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \vdash \Gamma, A\end{array} \qquad \begin{array}{c}\Pi_2\\ \vdots\\ \vdash \Delta, B\end{array}}{\vdash \Gamma, \Delta, A \otimes B}\otimes$$

then $\mathcal{N}(\Pi)$ is



where A_x and B_x are the A and B exit nodes of $\mathcal{N}(\Pi_1)$ and $\mathcal{N}(\Pi_2)$ respectively.

We can illustrate this conversion with a one-sided proof of $C, A \otimes B \vdash (A \otimes B) \otimes C$. One such proof is

$$\dfrac{\dfrac{\dfrac{\vdash A, A^{\perp} \qquad \vdash B, B^{\perp}}{\vdash A \otimes B, A^{\perp}, B^{\perp}}\otimes}{\vdash A \otimes B, A^{\perp} \parr B^{\perp}}\parr \qquad \vdash C, C^{\perp}}{(A \otimes B) \otimes C, A^{\perp} \parr B^{\perp}, C^{\perp}}\otimes$$

Conversion to a proof net proceeds as follows, working downwards from the axioms:

TO COMPLETE

This yields exactly the proof net from the previous subsection.

A crucial point to note is that there is a second one sided proof, differing only in the order of rule application:

$$\dfrac{\dfrac{\dfrac{\vdash A, A^\perp \qquad \vdash B, B^\perp}{\vdash A \otimes B, A^\perp, B^\perp}\otimes \qquad \vdash C, C^\perp}{(A \otimes B) \otimes C, A^\perp, B^\perp, C^\perp,}\otimes}{(A \otimes B) \otimes C, A^\perp \parr B^\perp, C^\perp}\parr$$

This alternate proof maps onto exactly the same proof net.

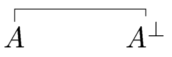
The way that the above two sequent proofs can map onto the same proof net illustrates the parallelizing nature of proof nets. Proof search for the sequent calculus is sequentially driven by the recursive nesting of the connectives in the formulas of the sequent. Proof nets unfold this recursive structure. The atomic leaves of the formulas are laid out for potential parallel access. Proof search with proof nets amounts to finding alternative ways of connecting up atoms, and this is only minimally constrained by the recursive structure of the formulas.

4.2 Classical Proof Nets

We now gives a more formal definition of proof nets for classical multiplicative linear logic, CMLL (without identities or exponentials).

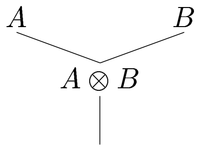
Links The links that can be used in a proof structure / proof net for CMLL are:

- Axiom link



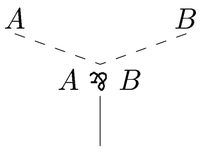
Premise (entry) nodes: none
Conclusion (exit) nodes: A, $A^\perp$

- Tensor link



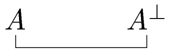
Premise (entry) nodes: A, B
Conclusion (exit) nodes: $A \otimes B$

- Par link



Premise (entry) nodes: A, B
Conclusion (exit) nodes: $A \parr B$

- Cut link



Premise (entry) nodes: A, $A^{\perp}$
Conclusion (exit) nodes: none

We will say more about cut links later.

Proof Structures A proof structure is a graph made up of links such that

1. Each premises of each link is connected to exactly one conclusion of some link
2. Any conclusion of any link is connected to at most one premise of some link

Proof Nets A switch graph (or switching) of a proof structure is a graph obtained by omitting one of two soft edges of of every par-link. A proof structure is a proof net if every switching is a tree.

4.2.1 Checking Correctness

The coorectness criterion for proof nets in terms of switching is also known as the Danos-Regnier criterion. Girdard originally used a more complex (but equivalent) criterion known as the long trip condition. Essentially, this says that any trip (cycle) from a node back to itself must pass through at least one par node. A wide variety of further alternative correctness criteria have been proposed

- Long trip
- Switching (Danos-Regnier)
- Acyclic-connected
- Hereditary secessiveness

- Graph parsing
- Deadlock freeness

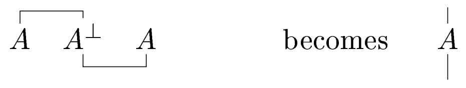
We will briefly review a quadratic way of checking the Danos-Regnier connection, due to Gallier. The naive algorithm exponentially constructs every switching and checks it for cycles. The Gallier algorithm (for cut free structures) is

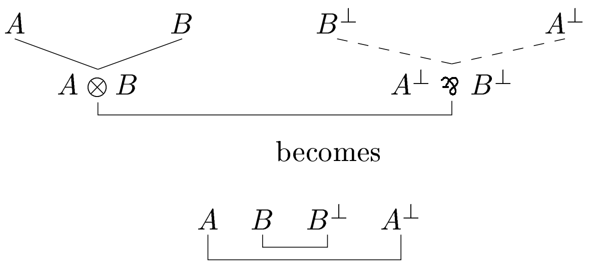
1. Base case:
   If the graph comprises a single axiom link, it is a proof net
2. Recursive 1:
   Delete all the bottom level par nodes (i.e. those whose conclusion nodes are not connected to anything), leaving their premise nodes in place. This has the effect of deleting all the bottom level soft edges. We then run the algorithm on the resulting (possibly unconnected) sub-graphs
3. Recursive 2:
   If there are no bottom level par nodes, we find a bottom level tensor node to delete. Deleting the node (i.e. removing the tensor link, but leaving its premises in place), must split the graph into two unconnected graphs $G1$ and $G2$. We then run the algorithm on $G1$ and $G2$.

   If removing the link results instead in a single connected (i.e. cyclic) graph, we must instead choose another tensor link to remove. That is, removing a tensor is a non-deterministic step. If no tensor is such that its removal splits the graph, fail.

4.3 Cut Elimination and Proof Nets

Mirroring the cut-elimination property of linear logic, cut links in proof nets can be eliminated. What is more, they can be eliminated directly and locally using the following reductions





[NOTE: rewrite to cover other symmetric cases]

4.4 Non-Commutative Proof Nets

Non-commutative linear logic (useful for categorial grammar) is obtained by dropping the rule of exchange. For proof nets, non-commuativity corresponds to a *planarity* condition

- In a *planar* proof net, axiom links must not cross one another

In order to ensure planarity, we must be careful about the left-right order of formulas when pushing negation inwards. Pushing negation in over either tensor or par flips the order of the conjuncts / disjuncts. Thus

$$\begin{array}{rcl}(A \otimes B)^{\perp} & = & B^{\perp} \parr A^{\perp} \\ (A \parr B)^{\perp} & = & B^{\perp} \otimes A^{\perp}\end{array}$$

Similarly, the individual formula trees in the proof frame must be set out in the order in which they occur.

Non-commutative proof nets have been applied to parsing categorial grammars. Morrill describes a scheme for tabularizing non-commutative nets (see chapter **??**).

4.5 Intuitionistic Proof Nets

Most linguistic applications of linear logic are in fact applications of intuitionistic linear logic. Intuitionistic in the technical sense that we are only interested in establishing a single conclusion (e.g. that a sentence is well-formed, or has a meaning).

4.5.1 Implication-Only ILL Nets

4.5.2 Essential Nets

Extension to intuitionistic implication-tensor fragment

4.6 Term Assignments for Intuitionistic Proof Nets

4.6.1 Implication-Only ILL Nets

4.6.2 ILL Nets

[Check what happens when we introduce tensor.]

4.7 Exponentials and Other Links

Chapter 5

Glue Semantics

This chapter gives an overview of ‘glue semantics’. This is an approach to the semantic interpretation of natural language that uses a fragment of linear logic as a deductive glue for combining together the meanings of words and phrases in a syntactically analysed sentence. For a recent collection of papers, see [Dalrymple99].

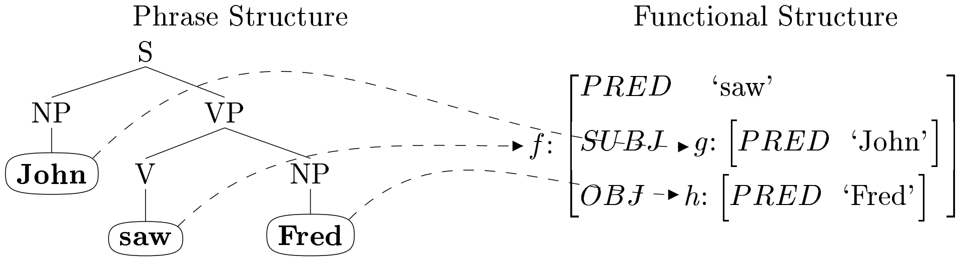
In its more recent developments, glue semantics bears an affinity to categorial semantics. That is, it is akin to the kind of compositional semantics obtained for categorial grammars by means of the Curry-Howard Isomorphism [Carpenter,Morrill]. A crude characterisation would be that glue semantics is like categorial grammar and it semantics, but without the categorial grammar. It provides a means for grafting a categorial style of semantics onto other syntactic formalisms; specifically Lexical Functional Grammar.

As we will see in the next chapter, linear logic has been extensively applied to categorial grammar. However, this has necessitated logical extensions to produce non-commutative and multi-modal versions of linear logics. These extensions are required to account for word order phenomena, which are one of the central concerns of any syntactic theory. By the contrast, the fragment of linear logic employed in glue semantics requires no new logical developments. Word order phenomena are assumed already to be accounted for by the background syntactic theory. The semantics just uses a conservative fragment of the kind of commutative linear logic reviewed in Chapter 3.

This is why we are starting our survey of linguistic application of linear logic with glue semantics: it requires no extra developments to linear logic. However, it does raise some algorithmic / proof search issues that do not arise so prominently in standard logical investigations. In particular, there is a problem of efficiently performing multiple derivations of a given conclusion from a set of premises. This corresponds to efficiently calculating all possible interpretations of a sentence because of the wholesale ambiguity that characterizes natural language.

5.1 Basic Glue Semantics

Let us introduce glue semantics by considering a very simple example, the sentence "*John saw Fred.*" Simplifying a number of syntactic details, we might present the syntactic analysis of the sentence as follows



The structures shown are what you would see in a Lexical Functional Grammar (LFG) style of analysis. The phrase structure (known as constituent- or c-structure in LFG) represents the kind of parse tree familiar from e.g. context free grammar. The functional structure (f-structure) is a projection off the c-structure that exhibits the traditional grammatical relations of 'subject', 'object', etc. The arrows show how nodes in the c-structure correspond to nodes in the f-structure. Note that that the Verb (V), Verb Phrase (VP) and Sentence (S) nodes in c-structure all correspond to the same f-structure node, here given the arbitrary label f.

We will assume that f-structure is is the primary syntactic structure determining compositional semantic interpretation. This assumption is a matter of convenience rather than necessity. Within versions of glue semantics developed for LFG, it is possible for structures besides f-structure to contribute to semantic interpreation (e.g. c-structure). And moving outside an LFG framework it should be possible to drive semantics directly from phrase structure.[1]

F-structure drives the semantics in conjunction with semantic entries in a lexicon. Let us use the meta-variable $\uparrow$ to refer to the f-structure node that a particular lexical leaf in c-structure projects onto. (If we were running direct off phrase structure, we would want $\uparrow$ to refer to the node that is maximal project of the lexical item. In the case of the V node, this would be the S node, and the two NP nodes are their own maximal projections.) Entries in the semantic part of the lexicon look like the following:

Word	*Meaning*	*Glue*
John	john	$\uparrow$
Fred	fred	$\uparrow$
saw	$\lambda y.\lambda x.\,\mathsf{see}(x,y)$	$\uparrow .OBJ \multimap (\uparrow .SUBJ \multimap \uparrow)$

[1] No work has been done on providing glue semantics from grammatical formalisms besides LFG. But in the absence of evidence to the contrary, there seems no reason why versions could not also be developed for other formalisms, like Head Driven Phrase Structure Grammar (HPSG).

This says that (a) whatever f-node the word **John** projects onto, $\uparrow$, its meaning is the constant john, (b) the meaning of the node that **fred** projects onto is the constant fred, and (c) the meaning of **saw** is a two place predicate, $\lambda y.\lambda x.\ \mathsf{see}(x, y)$, which requires its object $\uparrow .OBJ$ and subject $\uparrow .SUBJ$ meanings as arguments to return the meaning of the f-node that the verb projects onto, $\uparrow$.

In the lexicon, the $\uparrow$ meta-variables are uninstantiated. Thus a word like **John** can occur as either subject or object, and it is only a particular parse that can tell us which. Given the parse of the sentence "*John saw Fred*", we can instantiate the lexical meta-variables to specific values, namely

Word	*Meaning*	*Glue*
John	john	$\uparrow := g$
Fred	fred	$\uparrow := h$
saw	$\lambda y.\lambda x.\ \mathsf{see}(x, y)$	$\uparrow .OBJ \multimap (\uparrow .SUBJ \multimap \uparrow)$ $:= f.OBJ \multimap (f.SUBJ \multimap f)$ $:= h \multimap (g \multimap f)$

(To determine the value of $f.OBJ$, we have to follow the OBJ path from the f node to see which node is its value — in this case h.)

As a result of this instantiation of general lexical entries, we get three lexical premises

$$\begin{array}{rcl} \mathsf{john} & : & g \\ \mathsf{fred} & : & h \\ \lambda y.\lambda x.\ \mathsf{see}(x, y) & : & h \multimap (g \multimap f) \end{array}$$

The expressions on the left of the colon are *meaning terms*, written in some chosen meaning representation language. The expressions on the right of the colons are linear logic formulas. The atomic propositions, f, g and h, corresponding to syntactic constituents, represent semantics resource that produce and consume meanings.

Given our discussion of the Curry-Howard isomorphism in previous chapters, we can draw an immediate connection between meaning terms and proof terms. Meaning terms are simply proof terms embued with some additional internal structure. In the conventional CHI, proof terms for premises / assumptions are just arbitrary constants or variables. But here, the lexicon assigns non-arbitrary terms, sometimes with additional structure, to the lexical premises. Nonetheless, the combination of the meaning terms to form larger terms follows exactly the same rules as for the ordinary CHI.

If we ignore the meaning terms for ther moment, we have three lexical premises: g, h and $h \multimap (g \multimap f)$. What we want to do is find a derivation that consumes these lexical premises to construct a semantic output for the entire sentence, f. In this case, the derivation is very simple:

$$\cfrac{\cfrac{h \multimap (g \multimap f) \qquad h}{g \multimap f}\multimap_{\mathcal{E}} \qquad g}{f}\multimap_{\mathcal{E}}$$

To construct the meaning term for f, we include the meaning terms for the premises, and combine them by means of the functional application that corresponds to the rules of implication elimination:

$$\cfrac{\cfrac{\lambda y.\lambda x.\ \mathsf{see}(x,y) : h \multimap (g \multimap f) \qquad \mathsf{fred} : h}{\lambda x.\ \mathsf{see}(x,\mathsf{fred}) : g \multimap f}\multimap_{\mathcal{E}} \qquad \mathsf{john} : g}{\mathsf{see}(\mathsf{john},\mathsf{fred}) : f}\multimap_{\mathcal{E}}$$

To ease readability we have performed internal β-reductions on the meaning terms. The unreduced meaning term constructed for f is

$$[\lambda y.\lambda x.\ \mathsf{see}(x,y)](\mathsf{fred})(\mathsf{john})$$

The additional reductions are due solely to the extra structure of the initial meaning terms assigned by the lexicon.

To summarise from this initial example: syntactic analysis and lexical lookup provides a collection of lexical premises. Each premise comprises a meaning term paired with a linear logic formula (or glue formulas). Atomic propositions in the glue formulas correspond to syntactic constituents discovered in parsing. A glue derivation attempts to establish

$$\Gamma \vdash \mathcal{M} : \sigma$$

where Γ represents the lexical premises, σ is the atomic proposition corresponding to the sentence as a whole, and $\mathcal{M}$ is the meaning term for the sentence constructed via the Curry-Howard isomorphism.

More generally, glue semantics assumes that two levels of logical representation are at work in the semantic interpretation of natural language:

1. Meaning logic:
   The logic used to represent the meanings of words, phrases and sentences.

   Ideally, one would like to allow for some modularity in the choice of meaning language: e.g. first order logic, higher order internsional logic, discourse representation theory, situation semantics, or whatever one's favourite meaning representation language is. Glue analyses have to date been developed for a Montagovian style of higher order intensional logic, and for compositional DRT.

2. Glue logic:
   The logic used to deductively specify how meanings for words and phrases are to be assembled. There are various possible choices for a glue logic. But some form of linear logic seems appropriate for accounting for the basic resource sensitivity of semantic interpretation: in general the meaning of each word and phrase should be used once and exactly once.

5.2 The Core Glue Logic

We now set out the core glue logic underlying glue semantics. We do this by pointing out that there are two different versions of glue semantics. An early version (1993-97) that uses a more expressive formalism, and a recent version (from 1997) that uses the more restricted formalism exemplified in the last section. The advantages of the recent, restricted version are (1) increased computational tractability, and (2) that pretty much all the linguistic analyses originally developed in the exressive formalism can be reframed in the restricted version. The restricted formalism also brings out more clearly the similarities and differences between glue semantics and the kind of categorial semantics outlined in [Morrill,Carpenter].

5.2.1 An (Over) Expressive Glue Formalism

Example Revisited

The restricted formalism for glue semantics enforces a sharp separation between the meaning logic (in which meaning terms are expressed), and the glue logic. The earlier, more expressive formalism mixes them together more. Let us illustrate by going over our example of "*John saw Fred*" in the earlier formalism.

First, we give the lexical premises the old and new styles

New	Old
$\mathsf{john} : g$	$g \leadsto \mathsf{john}$
$\mathsf{fred} : h$	$h \leadsto \mathsf{fred}$
$\lambda y.\lambda x.\ \mathsf{see}(x,y) : h \multimap (g \multimap f)$	$\forall y,x.\ h \leadsto y \multimap (g \leadsto x \multimap f \leadsto \mathsf{see}(x,y)$

The older notation uses the binary predicate symbol, $\leadsto$, that pairs glue atoms with meaning language expressions. Thus $g \leadsto \mathsf{john}$ can be read as saying that node/resource g is assigned the meaning john. The formula $\forall y,x.\ h \leadsto y \multimap (g \leadsto x \multimap f \leadsto \mathsf{see}(x,y)$ says that for any meanings y and x, if h means y, then if g means x, f means $\mathsf{see}(x,y)$.

The derivation to get a meaning for f proceeds as follows. First, universal instantiation gives

$$\frac{\textbf{saw} \vdash \forall y,x.\ h \leadsto y \multimap (g \leadsto x \multimap f \leadsto \mathsf{see}(x,y)}{\textbf{saw} \vdash h \leadsto Y \multimap \forall x.\ (g \leadsto x \multimap f \leadsto \mathsf{see}(x,Y)}$$

The antecedent of this, $h \leadsto Y$ matches the premise for 'Fred', $h \leadsto \mathsf{fred}$, subject to unifying the variable Y with the constant fred. This gives us

$$\textbf{saw, Fred} \vdash \forall x.\ (g \leadsto x \multimap f \leadsto \mathsf{see}(x,\mathsf{Fred})$$

Universal instantiation of the variable x to X, and the unification of $g \leadsto X$ with $g \leadsto \mathsf{john}$ then gives us

$$\textbf{John, saw, Fred} \vdash f \leadsto \mathsf{see(john,fred)}$$

Thus we get to the same meaning as in the previous section.

Note that in this style of representation, glue inference depends on being able to unify meaning expressions. This has two undesirable consequences.

1. Unification increases the complexity of searching for derivations. In the example given here, first order unification suffices. But for only slightly more complex examples higher-order unification (at the minimum, second-order matching) is required. In the worst cases, higher-order unification is undecidable.
2. Unification of meaning expressions constrains the range of possible derivations. That is valid pairings of consumers and producers — according to the linear logic glue — may be ruled out because of unification failure. That is, the meaning expressions act as labels on deductions, controlling the range of derivations.

The new style of glue representation guarantees that meanings do not constrain valid derivations. There is no need for higher order unification, either to construct meanings or to limit derivations. On the other hand, the older style is more expressive.

Defining the Mixed Glue Fragment

To be more precise, let us specify the older, mixed glue formalism. Mixed glue formulas, G, can be defined as

$$\begin{array}{rcll} G & ::= & S \rightsquigarrow_e M \mid S \rightsquigarrow_t M & \text{(basic literals)} \\ & \mid & G \otimes G & \\ & \mid & G \multimap G & \\ & \mid & \Pi\lambda M.\ G & \text{(Quantification over M terms)} \\ & \mid & \Pi\lambda S.\ G & \text{(Quantification over S terms)} \end{array}$$

There are two typed form of basic literal. The e (for 'entity') type, $S \rightsquigarrow_e M$ assigns a type e meaning expression (M term) to a structure term (S term). Structure terms are constants referring to f-structure nodes,[2] or variables over such constants. The M term is an expression in the whatever the chosen meaning language is. The type t (for 'truth value') literal is exactly the same, except the meaning expression is presumed to be of type t rather than type e.

Higher order universal quantification over meaning expressions is permitted. We use Π to represent the universal quantifier to bring out the higher-order commitment more clearly. Typically we abbreviate $\Pi\lambda X.\ G$ as $\forall X.\ P$. Universal quantification over S-terms is also permitted.

The language thus defined is a fragment of higher order multiplicative linear logic, as used by Saraswat and Lincoln for concurrent linear constraint programming. Its sequent proof rules are shown in figure 5.1 The rules allowing lambda reductions to be employed are

[2] More accurately, S terms refer to nodes in a semantic structure projected off f-structure — see [Dalrymple] for details.

$$\frac{}{A \vdash A}\,axiom$$

$$\frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\,cut$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,l \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma \vdash A \otimes B}\,r$$

$$\frac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\,l \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,r$$

$$\frac{\Gamma, Pt \vdash A}{\Gamma, \Pi P \vdash A}\,l \qquad \frac{\Gamma \vdash Py}{\Gamma \vdash \Pi P r}$$

$$\frac{\Gamma, A' \vdash B \qquad A' \rightarrow_{\lambda} A}{\Gamma, A \vdash B}\,\lambda l \qquad \frac{\Gamma, \vdash A' \qquad A' \rightarrow_{\lambda} A}{\Gamma \vdash A}\,\lambda r$$

Where $A' \rightarrow_{\lambda} A$ indicates that A' lambda reduces to A, and in the right rule for quantification y is not free in Γ

Figure 5.1: Proof Rules for higher-order linear logic

what brings in the need for higher-order unification.

Core Fragment of Mixed Glue

Despite the expressive richness of the mixed glue formalism, nearly all the analyses framed within it turned out to fall within a more restricted fragment identified in [DalGupLamSar]. Syntactically, the fragment is

Syntax of Core Fragment of Mixed Glue

$$\begin{array}{rcl}
\langle\text{S-Term}\rangle & ::= & \langle\text{e-term}\rangle \mid \langle\text{t-term}\rangle\ \langle\text{t-var}\rangle \\
\langle\text{meaning}\rangle & ::= & \langle\text{meaning-const}\rangle \\
 & \mid & \langle\text{meaning-var}\rangle \\
 & \mid & \langle\text{meaning}\rangle(\langle\text{meaning}\rangle,\ldots,\langle\text{meaning}\rangle) \\
\langle\text{formula}\rangle & ::= & \langle\text{S-Term}\rangle \rightsquigarrow \langle\text{meaning}\rangle \\
 & \mid & \forall\langle\text{t-var}\rangle.\langle\text{formula}\rangle \\
 & \mid & \forall\langle\text{meaning-var}\rangle.\langle\text{formula}\rangle_1 \ \multimap\ \langle\text{formula}\rangle_2 \\
 & & \quad\text{if } \mathit{generic}(\langle\text{meaning-var}\rangle,\langle\text{formula}\rangle_1)
\end{array}$$

This formulation has a number of properties. First, it pushes the e/t distinction on the structure terms. Second, it only allows quantification over type t structure constants. Third, only implication is used. Fourth, although meaning variables can range freely over meanings, the form that the quantification can take is limited.

The restrictions on quantification over meanings amounts to the following: (1) all quantification over meaning is universal; (2) every quantification over a meaning X is associated with an implication, $P \multimap Q$; (3) the genericity of P with respect to X means that any meanings in P can only be built from X and other meaning variables. These restrictions ensure that any quantification over a meaning can be associated with a lambda abstraction, as we will shortly see.

The genericity condition is defined as follows

1. $\mathit{generic}(M, g \rightsquigarrow M)$
2. $\mathit{generic}(M, \forall S.\ P)$ (where S is a structure var) if $\mathit{generic}(M, P)$.
3. $\mathit{generic}(M, \forall N.\ P \multimap Q)$ where (M is a meaning var) if $\mathit{generic}(N, P)$ and $\mathit{generic}(M(N), Q)$

The force of the genericity condition can be seen if we define projections onto the structures and meanings of the core mixed formulas as follows

Structure (or type) projection, τ

$\tau(g \rightsquigarrow \mathcal{M}) = g$
$\tau(\forall S.\ P) = \forall S.\tau(P)$ (S is structure var)
$\tau(\forall M.\ P \multimap Q) = \tau(P) \multimap \tau Q$ (M is meaning var)

Meaning projection, μ

$\mu(g \rightsquigarrow \mathcal{M}) = \mathcal{M}$
$\mu(\forall S.\ P) = \mu(P)$
$\mu(\forall M.P \multimap Q) = \lambda M.\mu(Q)$
— provided P is *generic* for M

Consider the following formula, which is generic in M:

$$\forall N.\ s \rightsquigarrow N \multimap r \rightsquigarrow M(N)$$

It has the meaning projection $\lambda N.M(N)$, which η-reduces to M. In general,

If *generic*(M, P) then $\mu(P) = M$
up to α-,β-,η-conversion

This ensures that quantification over meanings in the core mixed formalism corresponds to lambda abstraction in its meaning projection.

An important point to note about the meaning and type projections just defined is that they allow us to translate certain mixed glue formulas into the newer glue formalism. A mixed formula ϕ translates to $\mu(\phi) : \tau(\phi)$. That is, we just pair the meaning and type projections of phi. For example

ϕ	$\mu(\phi) : \tau(\phi)$
$g \rightsquigarrow \mathsf{john}$	$\mathsf{john} : g$
$h \rightsquigarrow \mathsf{fred}$	$\mathsf{fred} : h$
$\forall y, x.\ h \rightsquigarrow y \multimap (g \rightsquigarrow x \multimap f \rightsquigarrow \mathsf{see}(x, y)$	$\lambda y.\lambda x.\ \mathsf{see}(x, y) : h \multimap (g \multimap f)$

However, not all mixed glue formulas successfully receive a meaning projection. As a rather contrived example, the formula

$$\forall X.h \rightsquigarrow X \multimap (g \rightsquigarrow \mathsf{john} \multimap f \rightsquigarrow \mathsf{likes}(\mathsf{john}, X))$$

does not get a meaninng projection, since the embedded implication is not generic in X. But this is in any case a very perverse lexical premise: a verb that means “likes”, but only if its subject means “John”.

5.2.2 Core Glue Fragment

We now describe the core fragment of the unmixed glue formalism. We'll start by defining the language, which is essentially the implicational fragment of linear logic plus universal quantification over atomic propositions / type, along with proof/meaning terms.

$$\frac{}{M : T}\ \mathit{identity}$$

$$\frac{\Gamma, M : T \vdash N : S}{\Gamma \vdash \lambda M.N : T \multimap S}\ \multimap_{\mathcal{I}}$$

$$\frac{\Gamma \vdash M : T \multimap S \qquad \Delta \vdash N : T}{\Gamma, \Delta \vdash M(N) : S}\ \multimap_{\mathcal{E}}$$

$$\frac{\Gamma \vdash M : T}{\Gamma \vdash M : \forall R.T}\ \forall_{\mathcal{I}}(R \text{ new in } \Gamma)$$

$$\frac{\Gamma \vdash M : \forall R.T}{\Gamma \vdash M : T[S/R]}\ \forall_{\mathcal{E}}$$

Figure 5.2: Proof Rules for Core Glue Fragment

$$\begin{array}{rcl}
\langle\text{meaning}\rangle & ::= & \langle\text{meaning-const}\rangle \\
 & \mid & \langle\text{meaning-var}\rangle \\
 & \mid & \langle\text{meaning}\rangle(\langle\text{meaning}\rangle,\ldots,\langle\text{meaning}\rangle) \\
 & \mid & \lambda\langle\text{meaning-var}\rangle.\langle\text{meaning}\rangle \\
\langle\text{type}\rangle & ::= & \langle\text{e-term}\rangle \mid \langle\text{t-term}\rangle\ \langle\text{t-var}\rangle \\
 & \mid & \langle\text{type}\rangle \multimap \langle\text{type}\rangle \\
 & \mid & \forall\langle\text{t-var}\rangle_1.\langle\text{type}\rangle \\
\langle\text{glue}\rangle & ::= & \langle\text{meaning}\rangle{:}\langle\text{type}\rangle
\end{array}$$

We call the linear logic propositions 'types' because of the underlying Curry-Howard Isomorphism (CHI). The propositions specify the types of the meanings in that they indicate what other (types of) meaning they should combine with.

Note that we are restricting quantification over types to quantification over atomic types of sort t. Technically this makes the logic second-order, but it is a very mild form of quantification. We could also regard the types as constituting a propositional logic with limited propositional schema.

We will represent the proof rules in natural deduction format (since the whole point is to combine meaning terms via the Curry-Howard isomorphism. These are shown in figure 5.2. These rules are standard, except that normally the quantification rules would give rise to an abstraction or application in the proof terms. This is not necessary here.

[Give explanation in terms of limited version of system F]

Possible Extensions

The core fragment represents a trade-off between expressive power and computational tractability. A large number of complex linguistic phenomena can be handled within the implication plus limited quantification fragment. However,some phenomena appear to motivate expanding the core to allow

- Tensor
- Exponentials
- Less restricted quantification

See final section.

5.3 Ambiguity and Multiple Derivations

A crucial feature of natural language is widespread ambiguity. This is a design 'feature' rather than a bug: without ambiguity communication would become impossibly verbose. Human interpreters of language a very good at resolving ambiguity by exploiting a variety of knowledge sources.

Not all ambiguity occurs at the syntactic or lexical level. If we assume that the principle concern of a grammar is to account for well-formedness judgements, then there are syntactically and lexically unambiguous sentences that nontheless allow multiple interpretations. Quantifier scope ambiguities are a well-known example of this. A sentence like "*A professor interviewed every candidate*" has two readings: one where the same professor does all the interviewing, and another where different candidates may have been interviewed by different professors. But there are no good grounds for positing a syntactic or lexical ambiguity in the sentence that could give rise to these two interpretations. Let us therefore call this 'semantic ambiguity'

Recall that a glue derivation aims to establish

$$\Gamma \vdash \mathcal{M} : s$$

where Γ is a collection of lexical premises derived from a syntactically analysed sentence, s is the semantic resource corresponding to the sentence, and $\mathcal{M}$ is a meaning term derived for s. Semantic ambiguity arises when, from a given set of premises, multiple derivations are possible.

A Modifier Scope Ambiguity

As a slightly artificial example of multiple derivations, consider the nominal phrase "*alleged criminal from London.*" Assume the phrase has a syntactic structure like the following:[3]

[3] The (f-)structure presented here flattens out what would be a syntactic attachment ambiguity in a phrase structure tree:

$$f : \begin{bmatrix} PRED & \text{`criminal'} \\ MODS & \{ \begin{bmatrix}\text{`alleged'}\end{bmatrix}, \begin{bmatrix}\text{`from London'}\end{bmatrix} \} \end{bmatrix}$$

Assuming the following meaning constructors[4]

$$\begin{array}{lcl} \lambda x.\ \text{criminal}(x) & : & f \\ \lambda P.\ \text{alleged}(P) & : & f \multimap f \\ \lambda P \lambda x.\ \text{from}(lon, x) \wedge P(x) & : & f \multimap f \end{array}$$

there are two distinct (normal form) derivations

$$\frac{\lambda P \lambda x.\ \text{from}(lon, x) \wedge P(x) : f \multimap f \qquad \dfrac{\text{alleged} : f \multimap f \qquad \text{criminal} : f}{\text{alleged}(\text{criminal}) : f}\multimap E}{\lambda x.\ \text{from}(lon, x) \wedge \text{alleged}(\text{criminal})(x) : f}\multimap E$$

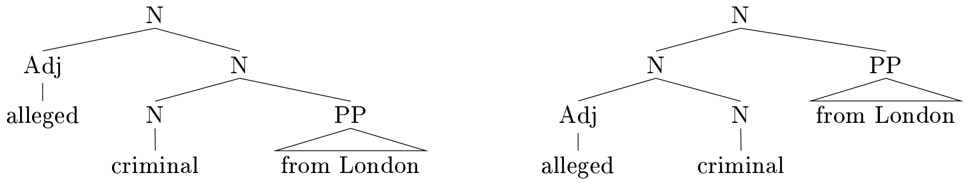
$$\frac{\text{alleged} : f \multimap f \qquad \dfrac{\lambda P \lambda x.\ \text{from}(lon, x) \wedge P(x) : f \multimap f \qquad \text{criminal} : f}{\lambda x.\ \text{from}(l, x) \wedge \text{criminal}(x) : f}\multimap E}{\text{alleged}(\lambda x.\ \text{from}(lon, x) \wedge \text{criminal}(x)) : f}\multimap E$$

In the first derivation, x is from London and alleged to be criminal. In the second derivation, x is not only alleged to be criminal, but also alleged to be from London. The two derivations arise as the result of different ways of permuting the $f \multimap f$ types to derive an f from an f.

5.3.1 Quantified NPs in Glue Semantics

Quantifier scope ambiguities are just an instance of the kind of modifier scope ambiguity illustrated above. However, quantified noun phrases have a slightly more complex type than the nominal modifier *alleged* and *from London*.

Let us consider the sentence "*Everyone saw someone.*" Assume the following f-structure and lexical premises:



[4] In fact, we would want to derive the constructor for "*from London*" from two lexical premises as follows:

$$\frac{\text{london} : g \qquad \lambda y \lambda P \lambda x.\ P(x) \wedge \text{from}(y, x) : g \multimap f \multimap f}{\lambda P \lambda x.\ \text{from}(\text{london}, x) \wedge P(x) : f \multimap f}$$

$$f_t\text{:}\left[\begin{array}{ll} \mathit{PRED} & \text{‘saw’} \\ \mathit{SUBJ} & g_e\text{:}\left[\begin{array}{ll}\mathit{PRED} & \text{‘everyone’}\end{array}\right] \\ \mathit{OBJ} & h_e\text{:}\left[\begin{array}{ll}\mathit{PRED} & \text{‘someone’}\end{array}\right] \end{array}\right]$$

- $\mathsf{everyone} : \forall S_t.\ (g_e \multimap S_t) \multimap S_t$
- $\mathsf{someone} : \forall T_t.\ (h_e \multimap T_t) \multimap T_t$
- $\lambda y, x.\mathsf{see}(x, y) : h_e \multimap (g_e \multimap f_t)$

Note that we have shown the semantic resources sorted according to whether they are sort e or sort t. The variables S_t and T_t range over constants or sort t. In this example there is only one such constant, f_t. We can therefore use the $\forall_{\mathcal{E}}$ rule to eliminate the quantifiers (leaving the meaning terms unchanged), to derive a simplified set of premises (with sorts no longer explicitly marked):

$\mathsf{everyone} : (g \multimap f) \multimap f$
$\mathsf{someone} : (h \multimap f) \multimap f$
$\lambda y, x.\mathsf{see}(x, y) : h \multimap (g \multimap f)$

There are two derivations of f from these premises, shown first without the meaning terms.

$$\dfrac{\dfrac{\dfrac{\dfrac{[h]^1 \quad h \multimap (g \multimap f)}{g \multimap f}{\scriptstyle\multimap_{\mathcal{E}}} \qquad (g \multimap f) \multimap f}{f}{\scriptstyle\multimap_{\mathcal{E}}}}{h \multimap f}{\scriptstyle\multimap_{\mathcal{I},1}} \qquad (h \multimap f) \multimap f}{f}{\scriptstyle\multimap_{\mathcal{E}}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[g]^2 \quad \dfrac{[h]^1 \quad h \multimap (g \multimap f)}{g \multimap f}{\scriptstyle\multimap_{\mathcal{E}}}}{f}{\scriptstyle\multimap_{\mathcal{E}}}}{h \multimap f}{\scriptstyle\multimap_{\mathcal{I},1}} \qquad (h \multimap f) \multimap f}{f}{\scriptstyle\multimap_{\mathcal{E}}}}{g \multimap f}{\scriptstyle\multimap_{\mathcal{I},2}} \qquad (g \multimap f) \multimap f}{f}{\scriptstyle\multimap_{\mathcal{E}}}$$

Including the meaning terms, we get the following for the first derivation (internal lambda reductions performed to increase readability).

$$\dfrac{\dfrac{\dfrac{\dfrac{[Y : h]^1 \quad \lambda y, x.\mathsf{see}(x, y) : h \multimap (g \multimap f)}{\lambda x.\mathsf{see}(x, Y) : g \multimap f}\multimap_{\mathcal{E}} \qquad \mathsf{everyone} : (g \multimap f) \multimap f}{\mathsf{everyone}(\lambda x.\mathsf{see}(x, Y)) : f}\multimap_{\mathcal{E}}}{\lambda Y.\mathsf{everyone}(\lambda x.\mathsf{see}(x, Y)) : h \multimap f}\multimap_{\mathcal{I},1} \qquad \mathsf{someone} : (h \multimap f) \multimap f}{\mathsf{someone}(\lambda Y.\mathsf{everyone}(\lambda x.\mathsf{see}(x, Y))) : f}\multimap_{\mathcal{E}}$$

The reader can verify that the second derivation yields

$$\mathsf{everyone}(\lambda X.\mathsf{someone}(\lambda Y.\mathsf{see}(X, Y)))$$

as a meaning term (after $\alpha\beta$-conversion).

Having seen some derivations, let us try to give the intuitions behind this treatment of quantified NPs. The sorts on the type $(g_e \multimap S_t) \multimap S_t$ reflect the familiar Montagovian type $((e,t),t)$ for quantifiers, which takes a one place predicate into a truth value.

In syntactic terms, the type $(g_e \multimap S_t) \multimap S_t$ says that the quantifier is looking for a constituent S of sort t which depends on the noun phrase g. The dependency may be because S directly subcategorizes for g (i.e. g is a required argument of S). Or the dependency may arise indirectly because S is constructed using some constituent that in turn subcategorizes for g. Having found a constituent S with an undischarged dependency on g (i.e. $g_e \multimap S_t$), the noun phrase discharges the constituent to discharge the dependency; it consumes $g_e \multimap S_t$ to give a modified version of the constituent S_t.

More semantically, the dependency on g introduces a variable (corresponding to an assumption in the derivation) at the point where g is subcategorized for. The variable is bound by the quantifier (discharged in the derivation) at S.

The fact that S is a variable means that a quantified noun phrase may potentially take scope over any constituent of sort t. However the dependency on g limits the number of constituents that can in fact form the scope of the quantifier. The limitations rule out precisely those scoping possibilities that would lead to either unbound variables or vacuous quantification in the semantic representations. That is, the range of possible glue derivations is sound and complete with respect to producing all possible quantifier scopings that do not involve unbound variables or vacuous quantification, as shown in [DalLamPerSar].

As an example of how unwanted scopings are prevented, we can consider a stock example from the literature on quantifier scope: "*Every representative of a company saw a sample.*" As pointed out in [HobbsShieber], rather than at least 6 scopings as the free permutation of the three quantifiers would suggest, this sentence only has five scopings. Permutations ruled out because of unbound variables (shown underlined) are

- $\forall r.\mathsf{rep\text{-}of}(r, \underline{c}) : \exists c.\mathsf{company}(c) : \exists s.\mathsf{sample}(s) : \mathsf{see}(r, s)$
- $\forall r.\mathsf{rep\text{-}of}(r, \underline{c}) : \exists s.\mathsf{sample}(s) : \exists c.\mathsf{company}(c) : \mathsf{see}(r, s)$

Permissible scopings are

- $\forall r.[\exists c.\mathsf{company}(c) : \mathsf{rep\text{-}of}(r,c)] : \exists s.\mathsf{sample}(s) : \mathsf{see}(r,s)$
- $\exists s.\mathsf{sample}(s) : \forall r.[\exists c.\mathsf{company}(c) : \mathsf{rep\text{-}of}(r,c)] : \mathsf{see}(r,s)$
- $\exists c.\mathsf{company}(c) : \forall r.\mathsf{rep\text{-}of}(r,c) : \exists s.\mathsf{sample}(s) : \mathsf{see}(r,s)$
- $\exists s.\mathsf{sample}(s) : \exists c.\mathsf{company}(c) : \forall r.\mathsf{rep\text{-}of}(r,c) : \mathsf{see}(r,s)$
- $\exists c.\mathsf{company}(c) : \exists s.\mathsf{sample}(s) : \forall r.\mathsf{rep\text{-}of}(r,c) : \mathsf{see}(r,s)$

To show how unwanted scopings are ruled out, consider the four partial glue derivations from lexical premises:

1. see(r,s): $$\frac{[r_e]^1 \quad [s_e]^2 \quad r_e \multimap s_e \multimap f_t}{f_t}$$

2. a-sample(s): $(s_e \multimap X_t) \multimap X_t$

3. a-rep(r)-of(c): $$\frac{\dfrac{[c_e]^3 \quad c_e \multimap ppn_t}{pp_t} \qquad ppn_t \multimap (r_e \multimap Y_t) \multimap Y_t}{(r_e \multimap Y_t) \multimap Y_t}$$

4. a-company(c): $(c_e \multimap Z_t) \multimap Z_t$

Note that there are two sort t constituents: the whole sentence f_t, and the prepositionally modified noun, “reprentative(r)-of(c)”, here identified as ppn_t. This means that the variables X_t, Y_t and Z_t could all potentially be instantiated to either f_t or ppn_t.

But for X_t and Y_t the only workable instantiation is f_t. This is because ppn_t cannot be made to have a dependency on r_e or c_e. Only f_t has this dependency, as shown by the assumptions of $[r_e]^1$ and $[s_e]^2$ in partial derivation 1.

For Z_t, one workable instantiation is ppn_t since this constituent directly depends on c_e. That is, the NP “*a company*” can take the nominal predication “*representative of _*” as its scope. But it is also possible to instantiate Z_t to f_t, so that “*a company*” takes the whole sentence as its scope. To see why, note that we can combine the partial derivations 1 and 3 as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{[r_e]^1 \quad [s_e]^2 \quad r_e \multimap s_e \multimap f_t}{f_t}}{r_e \multimap f_t}{\scriptstyle \multimap_{\mathcal{I},1}} \qquad \dfrac{\dfrac{[c_e]^3 \quad c_e \multimap ppn_t}{pp_t} \qquad ppn_t \multimap (r_e \multimap Y_t) \multimap Y_t}{(r_e \multimap Y_t) \multimap Y_t}}{f_t}\, Y = f}{c_e \multimap f_t}{\scriptstyle \multimap_{\mathcal{I},3}}$$

That is, by scoping "*every representative of _*" over the sentence f_t, a new dependency on c_e is acquired by f_t. This means that "*a company*" can then take scope over f_t.

The important point is that "*a company*" **must** take wide scope over "*every representative of _*", if both scope over the entire sentence f. For otherwise the necessary dependency of f on c won't be introduced. The two scopings that are bad because of unbound variables are exactly the ones that reverse this required scoping.

Summary

The treatment of quantified noun phrases in glue semantics is something that many people find initially hard to grasp; we hope that the preceding discussion aids rather than prevents understanding. One of the very significant advantages of the glue treatment of quantifier scope ambiguity, and more generally of modifier scope ambiguity, is the following. It does not require additional complex formal machinery, such as Cooper storage [**?**], to generate alternative scopings. Alternative scopings emerge directly from the basic mechanisms for semantic interpretation, without further stipulation. Moreover, it is not only quantifier scope ambiguities that are handled in this way. Other modifier scope ambiguities, like the "*alleged criminal from London*" example, or the scoping of negation and modals, are covered.

To be sure, the lack of additional scoping machinery is also an advantage shared by categorial grammar and its semantics. However, the much tighter correspondence between syntactic and semantic derivations in categorial grammar means that the sometimes inexact alignment between surface syntactic order and semantic scope can be problematic. This leads to, for example, the addition of an extra 'scope' connective to the Lambek calculus [Moortgat], or the further exploitation of multi-model or labelled deductive refinements orinally added to deal with certain word order phenomena (see next chapter). **[More discussion on this: here or later?]**

5.4 Structure Sharing Between Derivations

5.4.1 Ambiguity Managment

In this section we discuss a very significant computational issue raised by multiple derivations and ambiguity. We start with another favoured example of semanticists

> *Most politician can fool some of the people all of the time, a few politicians can fool all of the people some of the time, but no politicians can fool all of the people all of the time.*

This sentence is not unduly difficult to understand, but if one calculates the number of alternate quantifier and modifier scopings, the number of interpretations is astronomical. The precise number depends on particular assumptions about the semantic representation

— e.g. does it include quantification over events, etc. As an upper bound, assume 12 scope taking modifiers, all of which can permute freely only over the entire sentence. This leads to 12! or around 500 million readings.

This degree of ambiguity needs careful management, and is a major issue in computational linguistics. Three broad approaches to ambiguity management can be discerned:

Pruning Prune unlikely subderivations as early as possible to prevent the search space exploding. This is fine so long as one can measure likelihood on a purely local basis, so that one does not run into problems with local minima. Otherwise, there is a danger that a locally unlikely sub-derivation could form part of the most likely overall derivation. Pruning can lead to incompleteness: the correct / most likely derivation is never found.

Underspecification In semantics, underspecification often amounts to finding a spanning set of partial derivations, in much the way that was done for the "*Every representative of a company* example. Since combinatorial explosion results from different ways of putting these partial derivations together, this final step in the derivation is not taken. Typically a set of constraints is given limiting the number of ways the partial derivations can be assembled. Resolution amounts to tightening the constraints until only one or a small number of full derivations remain possible

Structure Sharing Structure Sharing is widely used in the management of syntactic ambiguity. The key observation is that even when there is massive ambiguity, there is a large common structure shared between the different derivations. Rather then recompute this common structure every time, it should be computed just once and shared across the derivations. In parsing, charts are often used to facilitate structure sharing. For context free grammars, they allow an exponential number of analyses to be computed in cubic time. In theorem proving, memoization or tabular deduction are the analogues of charts.

This section discusses how structure sharing can be used in connection with glue derivations. This is an area of active current research: a glue implementation operating in conjunction with Xerox's LFG grammar and parsing system (the XLE) is being run on sentences like the politicians example above.[5]

5.4.2 Memoisation for Proof Nets

Morrill's memoisation technique for planar proof nets. Applied to parsing categorial grammars. Crucial use of labelling within inference system. Difficulty of finding locally correct

[5] Currently, the prolog implementation of the glue semantics generates all the readings for the politicians sentence in well under half a second on a Sun Ultra 1. However, this result has yet to be properly confirmed: it has not yet been fully established that the algorithm is complete and really does produce *all* possible derivations.

sub-derivations without planarity restrictions. Unaware of any proposals for tabularisation of non-planar proof nets.

5.4.3 Chart-Based Techniques

Hepple describes a chart-based technique for deduction with the Horn clause fragment of implicational linear logic [Hepple]. This was originally developed for the purposes of categorial parsing, but applies directly to the task of glue derivation.

A Horn clause is an implication of the form

$$A_1 \multimap [A_2 \multimap \ldots (A_n \multimap B)]$$

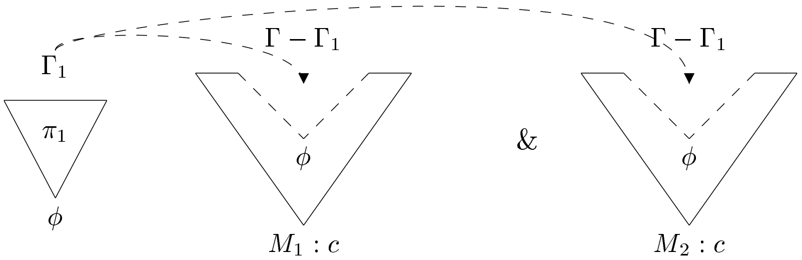
Thus both A and $A \multimap (B \multimap C)$ are Horn clauses, but $(A \multimap B) \multimap C$ is not a Horn clause.

Sharing and Packing in Charts

When alternative derivations are possible from a given set of premises, the derivations will typically have certain sub-derivations in common. The idea behind tabular methods is to record these common sub-derivations, so that they can be re-used in alternative wider derivations without having to be reconstructed from scratch each time.

There are two basic forms of tabularization. The first is *sharing.* Imagine that you are searching for derivations of some conclusion c from a collection of premises Γ. Suppose that you have found a sub-derivation, π_1, of ϕ from a sub-collection of premises Γ_1. Suppose there are also two or more sub-derivations of c from ϕ plus the remainder of the premises, $\Gamma - \Gamma_1$. Then the sub-derivation π_1 can be combined with all of these other subderivations to get a full derivation. This is shown pictorially below:

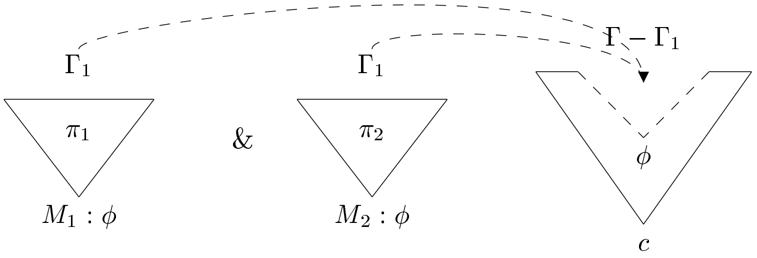
Sharing



Sharing slots one sub-derivation into several larger derivations. The flip side of structure sharing is *packing.* Packing slots several similar (but distinct) sub-derivations into one larger derivation, producing alternate versions of the larger derivation. Effectively the sub-derivations are packed together into an equivalence class — wherever one of the sub-derivations can be slotted in, any of the others could also be slotted in.

Suppose that you find two distinct proofs of ϕ from a given sub-collection of premises, Γ_1. Call these π_1 and π_2. Suppose that we also know that from $\Gamma - \Gamma_1$ and ϕ we can conclude c. Then either π_1 or π_2 can be slotted in to produce a deriavtion of c from Γ.

Packing



Sharing and packing can bring about substantial efficiency gains. For example, in parsing context free languages it allows one to uncover an exponential number of analyses in cubic time. For derivations in Horn clause linear logic [Hepple:???], it can find $n!$ derivations in 2^n time.[6]

Span Restrictions

In (!-free) linear logic, no premise can be used more than once. This means that if we have recorded a sub-derivation of $\Gamma_1 \vdash \phi$, we cannot slot it into another derivation $\Gamma_2, \phi \vdash \chi$ if Γ_1 and Γ_2 overlap.

This means that for sharing and packing, it is essential to record which 'span' (or sub-collection) of premises is used in a sub-derivation. Whenever two sub-derivations are combined, their spans must be disjoint, and the span of the resulting derivation is the union of the two sub-spans. When packing two sub-derivations together, it is essential they have identical spans.

(The same is true for charts in parsing. Here, however, a span is just a contiguous sequence of words in the sentence to be parsed. Disjointness of spans in parsing forces the following: two analyses of overlapping word sequences cannot be combined to form a single analysis of the combined sequence.)

Span disjointness and disjoint union for premise spans can efficiently be calculated by means of bit vectors. Recorded subderivation has an associated bit vector, with bits set for each premises used in constructing the derivation.

Horn Clause Compilation

The chart technique above only works for Horn clauses. However, the core glue fragment, though implicational, is not restricted to Horn clauses. For example, the instantiated type

[6] The difference between context free parsing and linear logic proof is that in parsing words / premises have to kept together in contiguous spans. But in the more general proof case, premises can be combined non-contiguously, so long as all premises are eventually used once and exactly once.

of a quantifier, $(n \multimap s) \multimap s$, is non-Horn.

Hepple presents a technique for the compilation of non-Horn clauses into (dependent) Horn clauses. Rather than present the details, we will illustrate the basic idea by means of an example. The method compiles the non-Horn formula

$$(a \multimap b) \multimap c$$

into two (Horn) formulas

$$a^i \qquad \text{and} \qquad b\{a^i\} \multimap c$$

where a^i is a (uniquely indexed) hypothesis excised from the non-Horn implication as a hypothesis; and $b\{a^i\}$ indicates a formula b whose derivation must make use of the excised hypothesis a^i. (Note that a is **not** a subformula of $b\{a^i\}$, in the way that a is a subformula of $a \multimap b$: it is merely an annotation on the formula b placing conditions on the way it is derived).

To illustrate, consider the inference $(a \multimap b) \multimap c, \;\; a \multimap b \;\vdash\; c$ with compiled Horn clauses. Compilation of the premises yields $a^i, \;\; b\{a^i\} \multimap c$ and $a \multimap b$. This allows a (Horn) derivation

$$\dfrac{\dfrac{a^1 \qquad a \multimap b}{b} \qquad\qquad b\{a^1\} \multimap c}{c}$$

Note how the derivation of b makes use of the hypothesis a^i, as required.

The basic idea behind the compilation method is that the rule of implication introduction

$$\dfrac{\begin{array}{c}[a]^i\\ \vdots\\ b\end{array}}{a \multimap b}\multimap_{\mathcal{I},i}$$

allows us to view and implication $a \multimap b$ and a derivation of b with an undischarged hypothesis / assumption of a. The compilation method drives this rule in reverse, excising a hypothesis of a from an implication $a \multimap b$. The indexing on the hypothesis is essential: we have to match the right hypothesis with the right implication.

After compilation, the rule of implication elimination needs to be revised

$$\dfrac{\begin{array}{c}[a]^i \ldots [d]^k\\ \vdots\\ b\end{array} \qquad\qquad b\{a^i, \ldots, d^k\} \multimap c}{c}$$

This rule necessitates a certain amount of book-keeping. Any formula needs to be marked with the set of undischarged excised hypotheses used to derive it — its index set. Showing these in braces before the formulas we could rewrite the elimination rule as

$$\frac{\{\phi\}B \qquad \{\psi\}(B\{i,\ldots,k\} \multimap C}{\{\phi \uplus \psi - i,\ldots,k\}C}$$

[term assignment for rule: follow Hepple or Lamping & Gupta: which is easier to explain?]

5.4.4 Skeletons and Modifiers

Modifiers are prevalent in natural language, both in syntax and semantics. Syntactically, an adjective like e.g. "*alleged*" modifies a noun to produce another noun. Semantically, it consumes a noun meaning and produces a modified version of it. At an intuitive level, adjectives thus have a logical type $N \multimap N$, where N is the type of a noun. In glue semantics, logical identities of this form signal the presence of a modifier.

To a logician, interested only in establishing whether a particular conclusion follows from a set of premises, logical identities corresponding to modifiers are just noise; throwing them away whenever they are encountered simplifies the search for a proof without affecting the type of the conclusion. To a semanticist, modifier identities are not noise. Although they do not affect the type of the conclusion, they *do* affect the meaning term assigned to it. Moreover, all ambiguities that are purely semantic in origin arise from different ways of inserting modifier identities into underlying skeletal (modifier-free) derivations.

[GuptaLamping98] describe a skeleton-modifier approach to glue derivation (i) initially perform a simple, 'noise free' derivation of a skeleton structure by leaving modifiers to one side whenever they are encountered, and then (ii) reintroduce the noise by inserting modifiers into the skeleton. Under suitable cirmcumstances, the initial derivation can be found deterministically and in linear time, with all the combinatoric blow-out being deferred to the second derivation stage.

As an example of an initial derivation, consider again "*alleged criminal from London*", with premises 1–4:

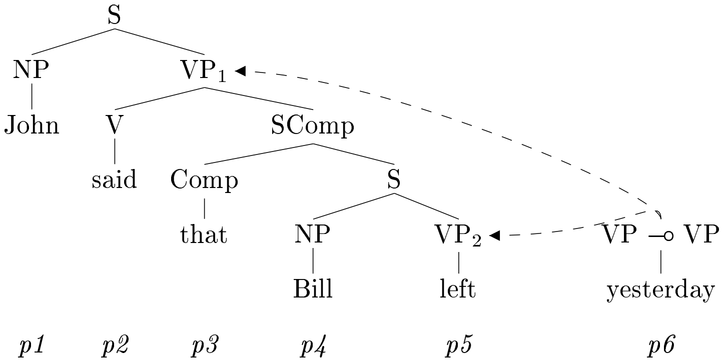
$$\begin{array}{lllccl}
1. & f \multimap f & \text{alleged} & \Longrightarrow & f \multimap f & \text{(modifier)} \\
2. & f & \text{criminal} & \Longrightarrow & f & \text{(skeleton)} \\
3. & g \multimap (f \multimap f) & \text{from} & \Longrightarrow & \underline{g \quad g \multimap f \multimap f} & \\
4. & g & \text{London} & & f \multimap f & \text{(modifier)}
\end{array}$$

Premise 1 is immediately held out as a modifier. Premises 3 and 4 derive another modifier, which gets held out, leaving premise 2 as the single remaining skeleton. Note that the result of combining premises 3 and 4 gives rise to a derivation tree with internal skeletal structure, whose conclusion is a modifier. A consequence of this skeletal structure inside modifiers is that modifiers can internally modify other modifiers.

Before giving a more precise definition of skeleton and modifier, it is worth considering whether Gupta and Lamping skeleton-modifier style of derivation is likely to be computationally advantageous.

Problems with Charts

Where modifiers are present charts miss significant possibilities for structure sharing. This can best be illustrated by an example from parsing. Consider "*John said that Fred left yesterday*", where either the saying or the leaving Verb Phrase can be modified by the VP $\multimap$ VP modifier "*yesterday*". If we are able to pull out modifiers from skeletons, this choice has a very simple representation:



Having obtained the skeleton parse tree, there is a simple two way choice about whether to insert the modifier high or low. Either way, the skeleton structure is completely shared between the two analyses.

But a chart-based approach precludes the sharing of this skeleton structure. Suppose that the chart first of all constructs the unmodified VP$_1$ "*said that Bill left*" spanning string positions $p2$–$p5$. This will lead to the memoization of the following intermediate constuents (string spans marked):

VP:	$p5$	NP:	$p4$
S:	$p4 - p5$	Comp:	$p3$
SComp:	$p3 - p5$	V:	$p2$
VP:	$p2 - p5$		

VP$_1$ can then be combined with the modifier "*yesterday*".

Not much of the work in building up VP$_1$ can be re-used in the analysis that attaches the modifier low, to VP$_2$. First VP$_2$ is combined with "*yesterday*", and a new VP with span $p5 - p6$ is recorded. Because the modified and unmodified versions of VP$_2$ have different string spans, we cannot re-use the S, and SComp constituents that we built before. Instead we must re-do the work building up from VP$_2$ to VP$_1$ via the S and SComp constituents. In doing so, the chart records the additional structures

VP: $p5 - p6$,   S: $p4 - p6$,   SComp: $p3 - p6$,   VP: $p2 - p6$

To summarize, the need to ensure span consistency can cause the chart to build shared skeleton structure multiple times. Localized differences in structure, namely the inclusion or not of a modifier at a particular point, lead to unnecessary global (span) distinctions being drawn. A skeleton-modifier approach to processing is appealing because it exploits kinds of structure sharing that chart-based approaches miss. It becomes doubly appealing if modifier insertion can employ further structure sharing.

Defining Skeleton and Modifier

At a first cut, a modifier is any formula equivalent to an identity $\phi \multimap \phi$. For example, $f \multimap f$, $(g \multimap f) \multimap (g \multimap f)$, and $g \multimap (g \multimap f) \multimap f$ are all modifiers.

GL's more precise definition of modifier first identifies skeleton and modifier occurrences of atomic propositions in a given formula. First convert the formula to negation normal by pushing negation inwards until it only applies to atoms.

Definition *If a formula in negation normal form contains one positive occurence of an atom A and one negative occurrence $A^{\perp}$, and the connective between the two subexpressions in which they occur is $⅋$, then they are modifier occurrences. All other occurrences are skeletal.*

Modifier occurrences come in pairs: a negative occurrence consuming a meaning, followed by a positive occurrence producing a modified meaning. The separation of the two occurrences by $⅋$ ensures that production follows consumption.

Separating Skeleton and Modifier: Horn Clause Compilation

A pure modifier is a formula that contains only modifier occurrences of atoms. Many lexical premises are impure, and contain a mixture of skeleton and modifier; e.g. $g \multimap (f \multimap f)$ and $(g \multimap f) \multimap f$. For the initial derivation stage to work, impure formulas need to be converted to a form $\mathcal{S} \multimap \mathcal{M}$, where $\mathcal{S}$ is pure skeleton, and $\mathcal{M}$ pure modifier.

In some cases, a simple equivalence-preserving re-ordering of antecedents achieves this effect. For example, $f \multimap (g \multimap f) \equiv g \multimap (f \multimap f)$, and $g \multimap ((g \multimap f) \multimap (h \multimap f)) \equiv h \multimap (g \multimap ((g \multimap f) \multimap f))$. If an implication is in Horn form — $a_1 \multimap (a_2 \multimap \ldots (a_j \multimap c) \ldots)$ — the existence of such a re-ordering is guaranteed since only c is positive, and a matching negative c can be swapped with a_j.

For some non-Horn formulas, no amount of rearrangement will place them in the form $\mathcal{S} \multimap \mathcal{M}$. The simplest example of a formula like this is $(g \multimap f) \multimap f$. GL use a partial application of Hepple's method of Horn clause compilation to deal with such cases.

Applied to a formula like $(g \multimap f) \multimap f$, Horn clause compilation produces two formulas

$$(g \multimap f) \multimap f \quad \Longrightarrow \quad g^i \quad \text{and} \quad f\{g^i\} \multimap f$$

where g^i is pure skeleton and $f\{g^i\} \multimap f$ is pure modifier. More generally, it is important that the Horn clause compliation is only applied partially, since compilation of a pure non-Horn modifier can excise a modifier atom and render the formula impure. Therefore rearrangement of antecedents and (recursive) Horn compilation is halted as soon as one obtains a pure modifier or a skeletal Horn implication into a pure modifier, even if the modifier is not Horn.

Having converted all the lexical premises into either pure skeletal Horn clauses, or pure skeletal Horn implications into modifiers, the first stage of glue derivation takes place. This matches positive and negative skeletal atoms, and holds any derived modifiers out to one side.

Skeletal Uniqueness Gupta and Lamping discuss a condition of skeletal uniqueness, whereby at most one positive and one negative skeletal occurrence of any atom is permitted within the entire lexical premise set. Skeletal uniqueness ensures that the initial deduction separating skeletons from modifiers is deterministic and linear time. Failure of uniqueness only affects the efficiency of the initial deduction process, and not its completeness. While skeletal uniqueness is mostly a linguistically natural restriction, it does fail for certain control constructions. But in these cases, the efficiency of the initial deduction is still not unduly harmed.

Non-Deterministic Modifier Insertion

GL discuss a non-deterministic approach to modifier insertion, since their aim is to stop after the initial stage of skeleton-modifier deduction. The simplest case of modifier insertion is when a derivation tree contains a sub-tree deriving ϕ, and the modifier to be inserted is (equivalent to) $\phi \multimap \phi$. The modifier can be adjoined to the sub-derivation, giving a new derivation also concluding in ϕ, e.g.

$$\begin{array}{c}\Gamma_1\\ \vdots\\ \phi\end{array} \qquad \begin{array}{c}\Gamma_2\\ \vdots\\ \phi \multimap \phi\end{array} \qquad \Longrightarrow \qquad \dfrac{\begin{array}{c}\Gamma_1\\ \vdots\\ \phi\end{array} \qquad \begin{array}{c}\Gamma_2\\ \vdots\\ \phi \multimap \phi\end{array}}{\phi}$$

The new derivation now has two occurrences of ϕ. Any further $\phi \multimap \phi$ modifiers can be inserted at either; i.e. either within the scope of the first modifier, or outscoping it.

However, modifiers of the form $\phi \multimap \phi$ can sometimes be inserted into derivation trees, even if the tree contains no sub-derivation of ϕ. We can (β-) expand derivations as follows

$$\begin{array}{c}\Gamma\\ \vdots\\ \phi\\ \vdots\\ \psi\end{array} \qquad \Longrightarrow \qquad \dfrac{\dfrac{\begin{array}{c}[\phi]^i\\ \vdots\\ \psi\end{array}}{\phi \multimap \psi}\multimap_{\mathcal{I}}, i \qquad \begin{array}{c}\Gamma\\ \vdots\\ \phi\end{array}}{\psi}\multimap_{\mathcal{E}}$$

in order to build up new sub-derivations. For example, we can build an insertion site for an $(h \multimap f) \multimap (h \multimap f)$ modifier in a tree with no such site as follows:

$$\dfrac{\dfrac{h \multimap (g \multimap f) \qquad h}{g \multimap f} \qquad g}{f} \quad \Longrightarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{h \multimap (g \multimap f) \qquad [h]^i}{g \multimap f} \qquad g}{f}}{h \multimap f}\,{\multimap_{\mathcal{I}i}} \qquad h}{f}\,{\multimap_{\mathcal{E}}}$$

For more complex modifiers, repeated applications of this expansion step extracting different hypotheses may be needed[7].

Packed Modifier Insertion

Work in progress

5.5 Discussion

5.5.1 Comparison with Categorial Semantics

5.5.2 Extending the Glue Fragment

Expressing scope constraints.

Using linear logic to model context update and dynamics.

F-structure paths as resources: dealing with reentrancy and resource re-use.

[7] The expansion introduces a (β) detour in the natural deduction proof, i.e. an introduction step immediately followed by an elimination. Inserting a modifier, or performing another expansion on the introduced implication,eliminates the detour. A more general expansion scheme to enable modifier insertion requires the introduction of η-detours (elimination followed by introduction) to break complex formulas into their component parts. However, the Horn form of the initial skeleton-modifier deduction ensures that all derivation trees are in η-long normal form; that is, every atomic proposition occurs as the conclusion of some sub-derivation. Hence η-expansion is not needed.

Chapter 6

Categorial Grammar

Chapter 7

Further Reading

Due to procrastination and poor time management, no bibliography has been compiled yet. The following are suggestions for further reading

Proof Theory I don't know of any good, really introductory books on proof theory. I benefitted a great deal from some unpublished notes of Gavin Bierman

- Troelstra & Schwichtenberg, *Basic Proof Theory*, Cambridge Tracts in Theoretical Computer Science, CUP

  Not all that basic, but comprehensive.

- Girard, Lafont and Taylor, *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, CUP

  Very readable — the table of contents looks forbidding, but the chapters on proof theory are very accessible.

Linear Logic & Proof Nets There are various introductions on the web, for example

- Pfenning
  `www.cs.cmu.edu/ fp/courses/linear/`

- Brauner
  `www.brics.dk/LS/96/6/BRICS-LS-96-6/BRICS-LS-96-6.html`

- Danlos & Di Cosimo
  `http://www.dmi.ens.fr/users/dicosmo/CourseNotes/LinLog/`

- Linear logic page
  `www.csl.sri.com/linear/sri-csl-ll.html`

- Bibliography available at
  `www.cs.cmu.edu/ carsten/linearbib/linear.bib`

In print

- Girard "Linear Logic" *Theoretical Computer Science* 50:1–102, 1987

  The original

- Girard "Linear Logic: its syntax and semantics", in Girard, Lafont & Regnier (eds) *Advances in Linear Logic*, CUP, 1995

- Troelstra *Lectures on Linear Logic*, CSLI Lecture notes 29, 1992

- Lafont, "From Proof Nets to Interaction Nets", in *Advances in Linear Logic*

- Lincoln, "Deciding provability of linear logic formulas" in *Advances in Linear Logic*

Linguistic applications For glue semantics see

- Dalrymple (ed) *Semantics and Syntax in Lexical Functional Grammar*, MIT Press, 1999.

For categorial grammar

- Morrill *Type Logical Grammar*, Kluwer 1994

- Moortgat "Categorial type logics" in van Benthem & ter Meulen *Handbook of Logic and Language*, Elsevier, 1997

- Carpenter *Type Logical Semantics*, MIT 1996

- Hepple "Memoisation for Glue Language Deduction and Categorial Parsing", Proc Coling-ACL 1998

  For chart-based deduction and Horn clause compilation

See also `http://esslli.let.uu.nl/Courses/retore-stabler.html` for resource logics and minimalism.