Contents

Chapter 1

Introduction

Chapter 2

Introduction

2.1 Our Intended Audience

These are notes for an introductory course on linear logic, 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 [**?**] 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:

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

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

then you probably meet the prerequisite.

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

- Categorial and type-logical grammar [**?**, **?**], including work on parsing categorial grammars [**?**, **?**], and the compositional semantics of categorial grammars [**?**, **?**]

- 'Glue semantics', which to a first approximation is a version of categorial semantics but without an associated categorial grammar [**?**]

- Resource-based reformulations of other grammatical theories, such as Minimalism Retore,Stabler, Lexical Functional Grammar [**?**, **?**] and Tree Adjoining Grammar [**?**]

- There have also been applications to such AI issues as the frame problem [**?**], 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.

2.2 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.

2.3 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

$$5 > 4, \quad 4 > 3$$
$$1 > 0, \quad 2 > 0$$
$$\forall i, j, k.\ (i > 0 \ \wedge \ j > k) \ \rightarrow \ (i \times j) > (i \times k)$$

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.

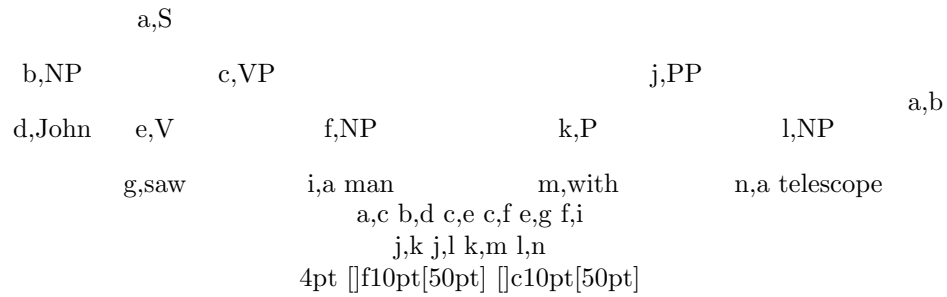
In linear logic formulas denote resources. Resources, unlike truths, get used up. A stock example of resource consumption is: (a) a packet of Gauloises costs 20FF, (b) a packet of Gitanes costs 20FF, therefore (c) if I have (a resource of) 20FF I can either buy a packet of Gauloises, or buy a packet 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

[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.

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:	$A, A \to B \vdash B$	
	$A, A \to B \vdash A \wedge B$	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

Traditional conjunction:	$A \wedge B \vdash A$	Can ignore B
Linear conjunction:	$A \otimes B \nvdash A$	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.

2.4 Proofs as Objects

2.4.1 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

[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.

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]:

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

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:

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

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:

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

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.

2.4.2 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

[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.

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

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

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

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

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

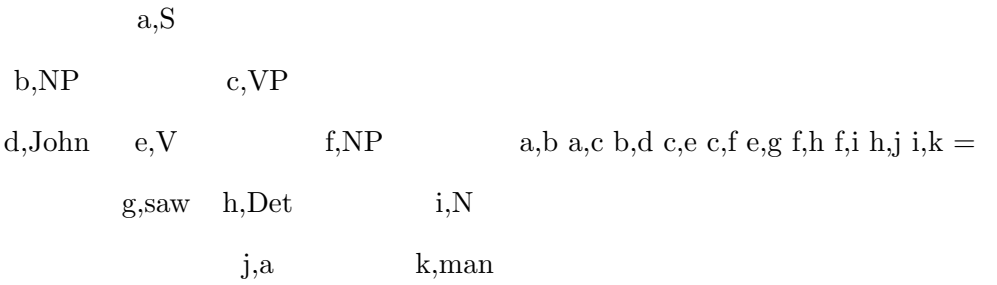
$$\begin{array}{llllll} {}_0\mathrm{NP}_1, & {}_1\mathrm{V}_2, & {}_2\mathrm{Det}_3, & \ldots, & {}_6\mathrm{N}_7 \\ \mathrm{John} & \mathrm{saw} & \mathrm{a} & \ldots & \mathrm{telescope} \end{array}$$

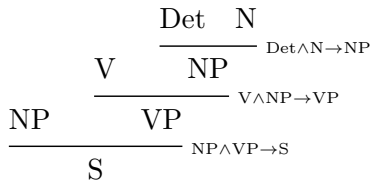
Letting Γ be the lexical premises plus the grammar rules, parsing becomes a search for a proof that $\Gamma \vdash {}_0\mathrm{S}_7$. What is interesting is not that ${}_0\mathrm{S}_7$ follows from Γ so much as there are two quite different ways of proving it. These two proofs correspond to the readings 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

[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.

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





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 **??**, a preliminary illustration is in order. Consider the standard natural deduction elimination and introduction rules for implication:

$$\frac{A \to B \qquad A}{B} \to_E \qquad\qquad \frac{\begin{array}{c}[A]^i \\ results B\end{array}}{A \to B} \to_I, i$$

The elimination rule $\to_E$ is just *modus ponens*, from A and $A \to B$, conclude B. The introduction rule $\to_I$ 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*:

$$\frac{f : A \to B \qquad a : A}{f(a) : B} \to_E \qquad \frac{\begin{array}{c}[x : A]^i \\ results f(x) : B\end{array}}{\lambda x.f(x) : A \to B} \to_I, i$$

We should think about the rule for *modus ponens* / $\to_E$ 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, $\to_I$, 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 $A \to B$.

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 $A \to B$ you can conclude B. The first proof (12) is sensible, the second (12) is pointlessly complicated:

Example 1.

Subexample 2.

$$\frac{A \to B \qquad A}{B} \to_E$$

[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.

$$\dfrac{A \qquad \dfrac{\dfrac{[A]^i \qquad A \to B}{B} \to_E}{A \to B} \to_I, i}{B} \to_E$$

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

Example 3.

Subexample 4.

$$\dfrac{f : A \to B \qquad a : A}{f(a) : B} \to_E$$

$$\dfrac{a : A \qquad \dfrac{\dfrac{[x : A]^i \qquad f : A \to B}{f(x) : B} \to_E}{\lambda x.f(x) : A \to B} \to_I, i}{(\lambda x.f(x))(a) : B} \to_E$$

Given the standard lambda-calculus rules of β- and η-conversion[6] note how the proof terms for (34) and (34) are equivalent,

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

indicating a semantic equivalence between the two derivations. Chapter **??** 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 (12) is the normal-form version of derivation (12).

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 two-fold. 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

[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

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.

2.5 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 **??** 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 $lwith$) and two forms of disjunction ($\parr$ and $lplus$). To go with these, it also has two forms of true ($\top$ and $lone$), and two forms of false ($\bot$ and $lzero$). 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: $.^{\bot}$
  $A^{\bot}$ 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 $A^{\bot}$ to leave nothing at all. Negation of a consumer gives rise to a producer, and vice versa, so that $A^{\bot\bot} \equiv A$.

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

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

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

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

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	$(SouplwithSalad)$
	$\otimes$
Fruit or cheese (depending on availability)	$(FruitlplusCheese)$
	$\otimes$
Coffee (free refills)	$Coffee$ $!Coffee]$

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

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

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