Chapter 2: The Language of Thought and UG: A Minimalist Combinatorial Categorial Grammar

Annabel Cormack and Neil Smith

1 Fragestellung

Given our commitment to CCG, it is incumbent upon us to spell out our presuppositions about the properties of UG that underpin explanations within that framework. Relevant phenomena must include Natural Language (NL) compositionality, such principles as structure dependence, the inclusion of combinators as part of the syntax as well as of the semantics, and the status of morphosyntax (MS) as the locus of all further generalizations and variation in the mapping from LF to PF in Natural Languages.

To begin, we raise and provisionally answer four related questions. The first is why we adopt a (bottom-up) Categorial Grammar rather than any of the alternatives: a top-down, Phrase Structure Grammar, or some different dependency format, or a Spec/Head/Complement based structure. The second is why we adopt and adapt Combinatorial Categorial Grammar (CCG) rather than relying on the traditional standard variety. The third is how and why we adopt a unidirectional CCG (i.e. one with no ordering parameters expressible) rather than a standard bidirectional one. The fourth is why combinators are part of the lexicon, rather than distinct forms of composition.

We do not attempt to justify our decisions in this brief preamble which is intended merely to provide an initial rationale for what follows. Detailed justification will appear in the text.

Q1. *Why Categorial Grammar?*

We have a number of reasons, listed from A1 to A6 below:

A 1. It unifies the Syntax and Semantics

Each syntactic category corresponds to a semantic type with respect to the number of arguments or operands it selects. Merge is simply function-argument application, operating in parallel in all of: the Language of Thought type system, the syntactic category system and the semantics. That is, this syntax is purely applicative, embodying a uniform computational system. This guarantees compositionality and should be attractive to philosophers and logicians as well as linguists.

A2. It is lexically based in that CG reduces the syntax to the manipulation of lexical features under Merge.

There is a converging view that all linguistic variation is lexical, and that most of the functions previously described by PS rules should be captured in the lexicon.

Crucially, the lexicon, including morphosyntactic features, encodes both UG facts and language-particular specifications.

A3. It is radically minimalist in the set of (primitive) categories postulated.

A4. It is monotonic. Although a phrase may be unpronounced, as in ellipsis, there is no deletion of LF interpretable material, and no movement, nor any syntactic 'internal merge'.

A5 It is monostratal. LF and MSyn are the interface representations, derived directly by Merge, ultimately of items from the appropriate lexicon. LF is mapped almost 1:1 onto LoT at the CI interface. MSyn is mapped onto sound, sign or writing at the S-M interface, via appropriate mapping rules.

A6. It is empirically superior. Evidence appears throughout the book, but a simple example is provided by the unproblematic characterisation of 'adjunct' that the system offers.

Q2. *Why Combinatorial Categorial Grammar?*

Again we have a number of reasons, listed from A7 to A12 below:

A7. According to Steedman (2000a:202) combinators have the advantage of being 'cognitively primitive'. This hypothesized meta-theoretical advantage is supplemented by a number of empirical bonuses:

A8. It provides for non-standard constituents with appropriate prosodic properties, as required for example for non-constituent conjunction and final contrastive focus.

A9. It can provide directly structures such as scope-affecting scrambling that would otherwise require LF displacement.

A10. '*Wh*-movement' can be eliminated.

A11. It allows the description of 'A-movement' without the need for movement as such.

A12. It avoids the problems of bound variables, or indeed any variables at all.

Q3 *Why a unidirectional CCG with morphosyntactically driven displacement?*

Again we have a number of reasons, listed from A13 to A15 below:

A13. It gives conceptual priority to LF. This allows LF to be directly syntactically generated and isomorphic to LoT representations with universal syntax.

A14. A unidirectional CCG allows the simplest Natural Language linearization, with at most a single choice: Functor First or Functor last. These alternatives do not have the linearization problems of analyses based on Kayne 1994.

A15. Aside from the lexical inventory, that part of parametric and microparametric variation among languages not accounted for by combinator choices or Functor First/Last is reduced to the morphosyntactic component of the lexicon. This includes 'displacement' ('head movement', phrasal movement) not accounted for with combinators. The parameters are cast as syntactically active default specifications; for example a verb in English is required to be in a checking relation with some head providing inflection.

Q4 *Why are combinators lexical items and hence part of the representation of syntax?*

Again we have a number of reasons, listed from A16 to A21below:

A16. Combinators are required in LoT to ensure adequate expressive power without ambiguity. Having the combinators appear in the syntax of NL as well ensures that it is unambiguous as to its structure and interpretation in LoT.

A17. As lexical items, combinators may be associated with morphosyntactic features, contributing inter alia to the elimination of the notion 'Specifier'

A18. The Polish notation associated with combinators allows the tree structure for an LoT expression induced by Merge to be represented unambiguously as a linear string, without the need to introduce brackets. This allows a simple mapping to a linearized LF.

A19. Some of the parametric variation in a language is mediated by means of the combinators, requiring their presence in the syntactic representation (e.g. to allow for checking for the appropriate combinator by a Raising head).

A20 They explicitly represent a particular structure and all the information relating to its well-formedness within a linear representation, including some non-standard constituents, so that processes depending on such constituency (e.g. ellipsis or some 'displacement') can be morphosyntactically characterised.

A21 The restrictive theory of NL and LoT combinators required to give a bracket-free notation automatically explains the existence of 'strong' extraction islands.

We return to all these putative advantages below and in the chapters to follow. Importantly, all of the properties in these answers should simplify the acquisition process.

There are, of course, potential problems: the psychological plausibility of the constructs of CCG, the learnability of the postulated morphosyntactic features, the empirical validity of

particular analyses, but these problems are not unique to us and we leave discussion of them to the relevant chapters.

2 UG, the 'Language of Thought' and Categorial Grammar

We are going to take as a starting position for natural language syntax the notion initially proposed by Fodor 1975, and more recently by Chomsky 2010 that NL is made possible by the prior existence of an internal capacity for mental representations involving recursion — the 'Language of Thought' (LoT) or 'mentalese', to use Fodor's (1975, 2008) terms.[1] We take it that LoT is both phylogenetically and ontogenetically prior to the development of NL, and there is now some experimental evidence that subjects construct an internal representation not derived from the native language (Culbertson et al 2012). More generally, we will argue for a fixed syntax for LoT with a vocabulary indicating that pre-linguistic infants and even some animals, from corvids to cuttlefish,[2] have elements of 'core knowledge' which underpin the representation of such properties as animacy, continuity, consistency, contact, having intentions, etc. (see e.g. Spelke 2004, Spelke & Kinzler 2007)[3], These examples are probably universal and innate; other LoT vocabulary may be culturally induced, and differ from person to person even in the same culture.

As indicated in Chapter 1, § 2.2, we further assume that LoT must be sufficiently like a natural language, e.g. in the arity of an item, to serve as a basis for the acquisition and use of the latter. We are going to argue that UG (the innate endowment that makes acquisition of a NL possible) provides this basis in the form of a Combinatory Categorial Grammar because the Language of Thought is so based. Additionally, we will argue that certain combinators are part of the lexicon of LoT.

We will first put forward arguments independent of NL for arity in LoT (§ 3.1), and then for an LoT that is linearized, allowing for a bracket-free, and variable-free notation (§3.2, §3.3). In order to give sufficient expressive power without variables, we argue that instead, LoT needs to include some combinators in its lexicon (§3.5). Combinatory logic has the same expressive power as the lambda calculus, does not require variables, and is standardly expressed in a bracket-free format.[4] So far as linearization is concerned, the simplest assumption is that LoT has the functor uniformly ordered with respect to its argument.

We assume that identical or comparable properties hold of NLs. We take it that a representation at LF is that of a particular meaning, which is a well-formed phrase of LoT.

---

[1] This position is close to that of Shaumyan in his Applicational Grammar (1977: 4ff), also using combinators, where the 'genotype language' provides an abstract model for the representation of thought.

[2] The somewhat surprising inclusion of cuttlefish is motivated by the report that they "keep track of what they have eaten, and where and how long ago they ate, in order to match their foraging behavior with the time of replenishing of different foods."(Jozet-Alves et al 2013).

[3] The relevant literature is too vast even to summarise, but see e.g. Abell et al 2000, Carey 2011, Smith 2001...

[4] It has been shown that just two combinators, for instance **S** and **K** , give the equivalent of the lambda calculus (see discussion and references in Steedman 2000a: Chapter 8).

The standard tool for unambiguous semantic representation of NL is usually taken to be predicate calculus, and when this proves too restricted, the lambda calculus.[5] These forms of representation if adopted as mental constructs involve two problematic elements: brackets, and variables. Any syntax or inference system including either of these is complicated, and neither has any obvious analogue in natural language.[6, 7] As they are not 'conceptually necessary' they should, if possible, be eliminated from the theory. Adopting a bracket-free but linearized Polish notation for LoT, and concomitantly for LF, solves this problem. Polish notation is 'Functor First' (section 3.2)[8] and, if every item is assigned a type, or, as we will see, a category in CCG notation, any well-formed representation is unambiguous (having a unique parse), and satisfies the desideratum that strings relevant to inference can be readily identified by pattern matching (as in section 3.4).

Combinators are available as items of NL too. That is, unlike standard CCG, we treat the combinators as lexical items both in NL and in LoT, so that the only Merge operation for LF is function-argument application. With suitable combinators, we can obtain many NL structures, and their LoT congeners, directly, including some of those standardly requiring 'displacement' or Move. Not all these 'displacements' can be synchronically ascribed to the necessities of externalisation, though each must once have been motivated by considerations of processing the external output.

Despite the priority we give to the Language of Thought vis-à-vis Natural Language, our position is also close to that of Hinzen 2013. He argues that there is no independent LoT, that thought and language are non-distinct cognitive domains, that syntax and semantics constitute "a *single* generative system" (p.11), and that 'narrow syntax' is a language of thought (p.10). This extreme position is close to that of e.g. Smith (1983:12) or Carruthers (1996) that we think in the LF of our native language, though it seems that not even all conscious thought is necessarily verbalised (recall the discussion of colour in Chapter 1 §2.2). We also take it that, in adulthood as well as infancy, the lexicon of an individual's LoT may be considerably more extensive than that subset giving rise to an LF that can map onto a well formed phrase in his NL: for instance, in the representation of different flavours. We then take an LF to be in its essentials a phrase of LoT, with identical syntax (merge rules). Contra Hinzen, and the strongest Minimalist position, we argue that every NL requires additional syntactic rules for well-formedness.

---

[5] These languages are taken to be the meta-language for describing truth conditions, but usually not to have any cognitive reality.

[6] Intonation, stress, timing and so on provide some guide to bracketing, but not always unambiguously.

[7] Chomsky (1977/79: 165/6) considers logics with and without variables for the 'notation actually used in mental representations'. He suggests that certain generalisations, for example concerning traces, cannot be formulated in a logic without variables, and hence that classical logic is the one used in mental representations. Even Copy Theory requires variables for the semantic interpretation of the lowest position of a Copied quantified DNP. We will show that the equivalent of a 'trace' can be accommodated in a variable-free notation (chapter 7).

[8] As far as we know, the only person to have regularly written his papers using Functor First notation in preference to brackets was its inventor, Łukasiewicz (Łukasiewicz,1957: 78).

The fact that LoT is prior to NL does not preclude some influence of NL on LoT. One example is given by the phenomenon of unaccusativity, which is manifest in NL but almost certainly not in pre-linguistic LoT. We discuss this in section 6, together with other potential problems for the identification of LF with a phrase of LoT, as we go along.

3 The syntax and semantics of the Language of Thought

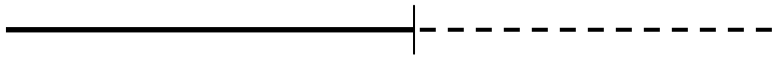
3.1 Desiderata for LoT

The first task is to determine what we can about LoT. We think there has been undue pessimism about the feasibility of this. A useful 'mental language' involves three components: a vocabulary, a syntax (rules for merge), and some means of relating a representation to the external world. It must also support inference. We have already postulated that the relation to the external world is indirect, being mediated by some internal model of the world (Chapter 1 §2.2). The vocabulary of LoT arises from the need to recognise recurring patterns of events in the world. Pre-linguistic children (at around 12 months) can recognise and evaluate goal-directed action (Gergely et al 1995). Thus 'rational Agent/initiator (of action)' and 'Goal state (of Agent)' are perceived by the infant as properties of some, but not all, unfolding events of motion over time by an entity. At the same age, the child knows that if he opens his hand when he is holding his teddy or biscuit, it falls to the ground. Here, the falling-event is not agentive with respect to the object falling, but rather either this event (or the state of the object being on the floor) was a goal of the child Agent. The child's thoughts involve at the least representations of two-place relations, where one of the places is taken by a motion event, one by the notion Agent, and the last by an Individual. The ascription of goals requires an Agent (a property) and a Goal, where the latter is internally complex, being in these examples a State given by an Entity and Location (a property). Thus there is some sort of compositionality. But these relations might be modelled non-linguistically, using an associative network of links, provided each unit has something approximating to a type, and an arity given in terms of type, to give the well-formedness conditions. For example, 'Agent' in a particular thought must usually be linked to Animate and Animate to some Entity with this property. Agent must further be linked to some Process, via a Causal relation, and Process to some Entity. Goal must be linked to Agent by some relations such as Want. Temporal and modal relations such as the unrealised nature of a Goal are perhaps implicit (not represented as such). It is clear that the underpinnings of a language are already established at this age. A very simple non-recursive 'grammar' could generate 'sentences' adequate to represent useful associations (e.g. that red fruit are good, or that green fruit are bad).[9] What is missing, if anything?

In the descriptions above, there are no transitive actions, and there is no linearization. Further, no node is linked to more than one item of the same type. It seems likely that these additional properties arise from generalisations about actions that realise goals. For example, in the dropping scenario above, the initiation of the hand-opening motion is prior to the start of the fall, so cause-effect relations are understood as temporally ordered. He may then

[9] Where a 'sentence' here consists maximally of an unordered set of a kind-name, a quality-name, and an evaluation, for instance.

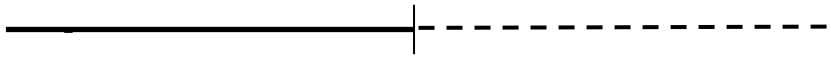
observe that Mummy achieves the same effect by the same means for something she is holding. The temporal situation is as in (1), where the dotted line represents an achieved goal, and the bold line, the cause or means of the agent obtaining that goal.

(1) 

(Agent) hand-opening culmination falling (non-Agent)

……event 1……> ……event 2……

The two simple events in (1), if co-occurring often enough, would plausibly be 'chunked' as a unit for more elaborated planning or reasoning. The structure in (2) is the result of such chunking and has the basic structure of a single complex event of the kind often lexicalised in NL (Moens and Steedman 1988). This new single event corresponds to LoT DROP2 (as for English agentive *drop*), which is encoded as a transitive relational head, with two entity arguments.[10]

(2) 

(Agent) PROCESS CULMINATION STATE (non-Agent)

……single structured event ……...

That is, the internal structure of the two events in (1), shown chunked in (2), is not spelled out overtly in the LoT DROP2. Rather, we take it that the existence of some agentive process causing the cessation of the holding or support of the object, its culmination, and the subsequent state of the falling of the object are all expressed in the Meaning Postulates for DROP2.[11] But if this is so, the syntax of LoT must distinguish the two arguments of DROP2. We take it that this is done by ordering the two arguments, with the non-agent associated more closely with the transitive relation DROP2, giving the standard <e, <e, t>> type with an external Agent argument (see further below).

The representations of LoT are semantically interpreted with respect to these internal worlds as in any model-theoretically interpreted language. The vocabulary will include at least names for entities, and items giving properties, and relations, and events relating to these entities. Additionally, a grammar that supports inference requires recursion. If some equivalent of 'P implies Q' is to be stored, we must already have recursion, since the whole is

[10] We use numerals with LoT items where the mnemonic English name would be unclear. Here DROP2 is intended to indicate a contrast with the non-agentive intransitive DROP.

[11] Meaning Postulates encode information about a lexical item relating to its use in inference (see section 6.2). Note that transitive DROP2 is not just a causative version of non-agentive intransitive DROP/FALL: you can't drop a plate by pushing it off the edge of a table. The Meaning Postulates must include reference to the agent's relaxing his grip on the object, where the grip may be indirect (e.g. with tongs).

of the same propositional kind as each of 'P' and 'Q', and the latter must each be well-formed complexes. We take it then as a working hypothesis that LoT is the simplest system with such properties. An indication that the version of LoT we formulate is on the right lines will be that it can map to and from the syntactic variety of NLs in natural and learnable ways.

LoT must be capable of representing conceptual structures in such a way as to be amenable to use for judgements of truth and for inference. The simplest useful sort of syntax for this purpose is one where concepts are combined compositionally and unambiguously to give representations of propositions whose truth value can be assessed with respect to the beliefs or evidence the thinker has. That is, LoT must have a vocabulary of basic concepts available to the thinker, and an ability to combine these to make representations of complex concepts such as propositions. For full compositionality, the items (simple or complex representations) must always be combined two at a time, entailing binary merge. The structures of LoT, then, are equivalent to binary branching trees.

The primitive merge in LoT should be the merge of pair of elementary items from the lexicon, resulting in a complex item, a phrase. A recursive rule will allow the merge of any two items, whether elementary or complex. Thus in LoT, given three elements from the lexicon, EVERY, DOG, and BARK, we should (abstracting away from number and tense) be able to merge EVERY with DOG, and EVERY DOG with BARK, giving EVERY DOG BARK. But it is also necessary that the representation can be used for inference, entailing that suitable strings represent states of affairs in some model. On grounds of simplicity we do not expect to find on drawing items from the LoT lexicon that instead of 'EVERY', we can only find 'λPλQ [EVERY P Q]]', with nine extra symbols. Nor do we expect some deus ex machina to enforce a condition that 'P IF Q' is not well-formed unless it is enclosed within a pair of brackets as '(P IF Q)', as is required in standard logic (e.g. Guttenplan 1986: 68, Rule Three). In other words, we apply to LoT the same principle of 'inclusiveness' (Chomsky1995a:228) as is applied in the Minimalist program to NL. There should be no syncategorematic symbols in an LoT (or, derivatively, LF) representation, and hence, no brackets, nor any equivalent tree representation. Polish notation, described below, satisfies this desideratum, provided each lexical item is given a fixed type. We argue that there are other cogent reasons for adopting Polish notation.

In order for concepts to combine in any interesting fashion, they cannot all be of the same kind. An (unordered) thought of 'redness, goodness, my-hungriness, my-ingestion, fruitness' may be useful for storing associations and extracting statistical correlations, but is not useful for making hypotheses or for logical inference. We have thoughts about entities and about properties of entities and relations between entities. To express all of these we use one standard notation.[12] At the simplest, propositions expressed in LoT have the type <t> as their

[12] This is the notation of for instance Dowty, Wall and Peters 1981, and Heim and Kratzer 1998. Under this notation, types are in angle brackets, and an item *f* of type <α, <β, γ>> can be merged with *g* of type <α> to yield a phrase *f g* of type <β, γ>, and then with *h* of type <β> to yield a phrase *f g h* of type γ. Only this order of merge is licensed. It would be more consistent to use a bracket-free Functor First notation here too, as Shaumyan (1977) does with prefixed Δ, but we use this version for the sake of familiarity, since nothing hangs on it.

category. A name referring to an entity, such as LAPIS (a particular cat) has type <e> (we use small capitals for items of LoT where these have corresponding LF congeners). If we conceive of a property like GREEN as representing a function from entities to truth values, then we may give GREEN the type <e, t>, the type for a function from type <e> to type <t>, to represent just this. Now we may form a complex concept by juxtaposing GREEN and LAPIS, interpreted as applying the function GREEN, to the entity LAPIS. The function GREEN returns a value T (true) or F (False) with respect to the current mental model of the world. 'GREEN LAPIS' has value False in the current world. Relations between entities are represented in Curried form, as having type <e, <e, t>>, rather than say <e, e> ∈ R, or <<e, e>, t>. This induces binary branching, since only one selection for a type <e> item is available at a time.[13] The <e> selections in <e, <e, t>> are ordered, and by convention, the outermost must be discharged first. The notation '<e, <e, t>>' encodes the same information as an ordered theta grid (Grimshaw, 1990), or as the Predicate Argument Structure of Rappaport and Levin (1988). However, instead of marking the external selection by underlining, it is encoded by being the innermost selection (selection 1); and in the <t> it also encodes the type of the entity resulting from saturating all the specified selections of the predicate.

Meaning Postulates (statements designed to restrict the denotation of one meaning in relation to others) associated with each item determine which selection for <e> corresponds to, for example, the 'agent/experiencer' or 'patient/theme' role or whatever is appropriate, and generalisations over the lexicon keep these aligned for different lexical items. In the discussion, we follow the usual convention that the innermost selection corresponds to the agent/experiencer, and the outer to the patient/theme, to avoid having to spell out the Meaning Postulates. In other words, if the conceptual system is enriched with the notion of arity, a useful way of representing complex propositions is offered, one which lends itself not just to memorising situations, but to remembering and manipulating articulated representations of hypothetical states of affairs. The types combine under function-argument application, so that their syntax is that of a categorial grammar with (for the moment) just two atomic categories, <e> and <t>.

The next question is whether LoT needs to be linearized, like NL. We argue that the answer is 'Yes'. This is contrary to Chomsky's (2013:36), 'projection' principle T:

(3) (T) Order and other arrangements are a peripheral part of language, related solely to externalization at the SM interface, where of course they are necessary.

We reject this principle. We will argue on grounds internal to LoT that it should be linearized. First, this allows for a representation that is unambiguous, but without brackets; and second, it is a desideratum for efficient inference. We consider first the argument for a bracket free

[13] Rizzi (2013a:5) raises the possibility that, because it would be equally possible to use a Relational notation, *R*(*x* ,*y*) rather than a Curried one in the semantics, the latter has to be given as an 'inherent formal property' of UG. In relational notation, Merge would require two distinct operations, one forming ordered sets (consisting of the appropriate number of suitable items), and one forming function-argument pairs (the head and its set-argument, where the cardinality of the set has to be determined as appropriate). Under Curried notation, only the latter is required, and Merge can be defined so that it operates recursively.

Polish notation, and in particular a Functor First notation, and then the argument for the facilitation of inference.

3.2 Linearization

We argued in Cormack and Smith 2005b in relation to natural language that if Merge is selection driven, then linearization too should be selection-driven. We argued that the LF of a natural language was linearized, by Functor First or Functor Last. Merge is selection driven in LoT too (by types), with the obvious implication. The simplest way of effecting linearization is to order the item whose selection is discharged — the functor in the merge — either as 'first' or as 'last'. This gives 'Functor First' or 'Functor Last', equivalent to 'Polish' (or 'prefix') notation, and 'Reverse Polish' notation respectively. In either case, the Merge rule must be function application (i.e. the grammar is purely applicative). Here, we argue on grounds independent of NL for the linearization of LoT.

Suppose we have a predicate FEED of type <e, <e, t>>, interpreted as an item from LoT. Suppose further that LoT supplies two further items, FIDO (a dog) and JERRY (a boy) of type <e>, and that what is to be expressed is that Jerry feeds the dog. The idea that items like FEED are representations of semantic functions leads us to expect that the three items can be merged and linearized in one of the six ways in (4).

(4) (a) [[FEED FIDO] JERRY] (b) [JERRY [FIDO FEED]] (c) [[FEED FIDO] JERRY]

(d) [[FIDO FEED] JERRY] (e) [JERRY [FEED FIDO]] (f) [[FIDO FEED] JERRY]

Current minimalist grammars usually take it that linearization is a property imposed on an unlinearized internal representation by NL, so that all the orders above are potentially available. Under the assumption of Functor First/Last, there are only two legitimate LoT forms for the merge of the three items with the types given:

(5) a Functor First: FEED FIDO JERRY (i.e. [[FEED FIDO] JERRY])

b Functor Last: JERRY FIDO FEED (i.e. [JERRY [FIDO FEED]])

Because we will suggest in section 4.1 that all NLs are temporally Functor First, we will for ease of comprehension present LoT merges with the functor on the left of its operand on the page, so that it is first as read.

Linearization under Functor First has two major advantages: it accommodates any sort of type or CCG category, including adjuncts and 'specifiers'; and it renders brackets redundant provided that the types for the items are known. Further, given a well-formed representation (e.g. drawn from memory), constituency, that is, which items are merged together by binary merge under function application, is unambiguous. As to the elimination of brackets, this simplifies the syntax of LF, since otherwise the left bracket and the right bracket have to be included in the lexicon of LoT, and an account given of the unbounded nested dependencies

between a left and a right bracket. Their role in inference would also need specification.[14] We show a few examples of linearization under Functor First. Let P and Q be propositions of type <t>, DOG and BARK one-place predicates of type <e, t>, THINK a predicate of type <t, <e, t>>, EVERY a binary quantifier of type <<e, t>, <<e, t>, t>>, and IF a connective of type <t, <t, t>>. Then abstracting from the content for each type (i.e. considering only grammaticality in LoT), Functor First induces the orders and constituency shown in (6), and no other orders or constituency other than those obtained by switching BARK and DOG, as the reader may verify.

(6) a IF P Q i.e. $[[\text{IF}_{<t, <t, t>>}\ P_{<t>}]_{<t, t>}\ Q_{<t>}\ ]_{<t>}$

b EVERY DOG BARK i.e. $[[\text{EVERY}_{<<e, t>, <<e, t>, t>>}\ \text{DOG}_{<e, t>}]_{<<e, t>, t>}\ \text{BARK}_{<e, t>}]_{<t>}$

c EVERY DOG THINK P i.e. $[[\text{EVERY DOG}]_{<<e, t>, t>}\ [\text{THINK}_{<t, <e, t>>}\ P_{<t>}]_{<e, t>}\ ]_{<t>}$

Note that the complex adjunct [IF P] is unproblematically linearized before its host Q in (6a); the quantifier phrase [EVERY DOG] appears preceding its predicate in (6b), and the clause P follows its selecting verb in (6c). The contrast between (6b) and (6c) in the placement of the non-subject argument of the verb shows that Functor First does not deliver either a 'head final' or a 'head initial' ordering in the verb phrase, but something better aligned with NL, where quantified subjects and objects are typically pre-verbal, and clauses are typically post-verbal. Importantly, this does not rely on any notion of 'specifier'.

3.3 Why 'Functor First' not 'Functor Last' linearization?

Mark Steedman (Steedman 2000a: 202, 2002) presents an argument for a human capacity to utilise combinators based on mental processes that were prior to communicative language. Specifically, he suggests "that the language faculty evolved in the species and develops in children by a rather direct adaptation of a more primitive apparatus for planning purposive action in the world by composing cognitive representations of operations on objects or tools.

Suppose this is right, and that someone standing on a rock gazing at a banana tree wants to get a banana. Then a plan to achieve the state of having a banana (state 1) might be planned by remembering that being up the tree (state 2) gives the possibility of being in state 1 by simply reaching for a banana (REACH). Further, standing under the tree (state 3) makes possible the climbing of the tree (CLIMB) to achieve state 2. A further action of moving to the tree (GOTREE) from state 4, being on the rock, gives the possibility of attaining state 3. The actions are functions from one state to another. A plan is a representation which organises selected actions in such a way as to achieve the goal state from the start state.

(7) Start state 4. Goal state: 1

REACH: state 2 → state1 ; GOTREE: state 4 → state 3; CLIMB: state 3 → state 2.

Plan 1: [**Compose** GOTREE CLIMB]: state 4 → state 2

[14] If all we were concerned with were truth conditions, it would be irrelevant what notation was used to represent meanings. But here, LF is intended to be a level of representation. If it includes brackets, then by the Inclusiveness Principle, these would have to occur in the LF associated with certain lexical items — they could not simply occur syncategorematically as in standard formal logic. The same applies to the disambiguating device of a tree diagram.

Plan 2: [**Compose** [**Compose** GOTREE$_{<4,\ 3>}$ CLIMB$_{<3,\ 2>}$] REACH$_{<2,\ 1>}$]: 4 → 1

Plan2 is a desirable plan. The representation could of course be given in Functor Right format, rather than Functor Left. But it cannot contain orders like '**Compose** CLIMB GOTREE', because this would amount to a 'backward' version of **Compose**, which is not an 'orderly' combinator (section 3.6). It follows that the activity to be carried out first in the temporal ordering must be adjacent to **Compose** in each representation, whether Functor Left or Functor Right (mirror image) is used. We surmise that this is the basis of a universal Functor First order of NL. We might indeed see the restriction to orderly combinators as an instance of Grice's (1975) 'Be orderly', applied to internal representations.

We turn now to the argument from inference.

3.4 The argument from inference

If some information is brought to my attention which I can encode as P, and I want to know what implications this has, then I need to be able to identify and use stored information pertaining to P. To use the relevant information in an automated fashion, at least one inference rule must be assumed. We assume that a human mind/brain sufficiently mature to form LoT representations can also utilise such an inference rule. The primary one relates to IF.

This information resulting from inference may be required immediately, so inferencing must be simple. In particular, I do not want to re-parse my stored representations to find out whether they include P, and further, whether they are such that P serves as the antecedent in some conditional. The information should be obtained by pattern-matching alone (i.e. testing for identity of unanalysed forms, without parsing). Pattern-matching is going to be easier if the representation is linearized and the relation of P to IF can be taken to be simple adjacency. The idea is that if you construct an LoT representation, and store it for later use, then given that it is unambiguous, you should be able to use it without parsing it. It is comparable to adding two large numbers without paying any heed to what the numbers are but just concentrating on the immediate digits. This immediately suggests that IF should be peripheral, so that a conditional can be immediately identified. We refer to this position as 'first' and represent it on the left. Functor First is provably sufficient to answers the desideratum for inference by pattern matching; we exemplify below (section 3.7). Accordingly, the inference rule (an axiom), is as given in (8), with IF first:

(8) IF *X Y*, *X* ⊢ *Y* for *X*, *Y* of type <t> and IF of type <t, <t, t>>

The operator IF is of type <t, <t, t>>. Inference rules always relate representations of propositions, having type <t>, so that *X* and *Y* must have type <t>. *X* and *Y* are 'unification variables'; they are placeholders for something of type <t>, and can unify with any entity of type <t> in the model. The inference rule is read as: if the two LoT representations (separated by commas) on the left of the turnstile '⊢' are taken to be true in some model, then *Y* may be taken to be true in that model. In order to make an inference from P, it is sufficient to pattern-match P against the string of symbols following IF. If this succeeds, then the remaining string represents the consequent.

By contrast, in standard infix notation, the inference rule is as in (9), where every constituent must be delimited by brackets, to avoid ambiguity.

(9) $((Y)\ (\text{IF}\ (X))),\ (X) \vdash (Y)$ for X, Y of type <t> and IF of type <t, <t, t>>

Given new information encoded as (P), the best strategy would be to count past exactly two right brackets on the right-hand end of the string, and then pattern match for (P). If this succeeds, look leftwards. If the next item is 'IF', and if the following item is a left bracket, then the remainder, omitting the leftmost left bracket, is the consequent. This seems to be a somewhat convoluted algorithm. The other more serious disadvantage is that it is not until after (P) has been pattern matched that it can be discovered whether the expression '((*Y*) (IF (*X*)))' is a conditional, rather than say '((*Y*) (OR (*X*)))'. In Functor First notation, any string not beginning with IF can be discarded immediately.

We emphasize that this will only work as described if all the representation are 'functor first' (or 'functor last') throughout. This result supports the assumption that linearized unambiguous 'functor first' bracket free notation is what LoT uses. This is what we will assume, not just for the relation between clauses and binary operators, but throughout.

As we observed above, the order given by 'Functor First', when written on the page with 'first' equating to 'leftmost' corresponds to Polish notation for the grammar. Which of two items being merged is the functor and which its argument can be read off the types. Because the only merge rule for types is function application, the functor must have type <α, β> and the argument, <α>, for some α and β. This facility will be preserved even when the combinators are introduced, because they are typed items of LoT.[15]

3.5 Meaning Postulates

Inference may depend not only on axioms, but on Meaning Postulates. These represent not just beliefs or hypotheses, but are statements 'deemed'[16] to be true in any model, including those of imaginary worlds where other facts true in this world might be suspended. For example, a human equipped only with the inference rule for IF in (8) might find it useful to have a new connective, represented as AND, which fulfils the two conditions in (10) (informally, the first reads 'IF [AND *P Q*] is true then *P* is true') :

(10) (i) IF AND *P Q P* is deemed to be true

(ii) IF AND *P Q Q* is deemed to be true

where 'AND' is of type <t, <t, t>>.

[15] We will introduce more details of 'events' and their type later but, despite having no proof, are reasonably convinced that a neo-Davidsonian syntax for LoT is implausible as the underpinning for NL, given that its syntax is so dissimilar to that of NL. Also, the usual neo-Davidsonian syntax requires brackets and variables for quantification (but see Champollion 2011).

[16] In the sense of Grice 1982:243.

The pair of Meaning Postulates thus functions to restrict the meaning of AND exactly to that appropriate to conjunction. Alternatively, of course, the meaning of AND is defined directly, by its inference rules:

(11) AND *P Q* ⊢ *P*; AND *P Q* ⊢ *Q*

It is possible that conjunction is given *ab initio*, but there are anyway various reasons why LoT might like such a connective. For example the representations in (12a) and (12b) (for *P*, *Q*, *R* of type <t>), are equivalent. But (12a) might be easier to store, because no item is repeated.

(12) a IF AND *P Q R* b IF *P* IF *Q R*

Alternatively, for the inferences likely to be needed, it might be convenient to have P and Q closely associated as operands of one head in the structure.

Meaning Postulates are often used to compress information, as for instance with the item BACHELOR defined via the MP in (13):

(13) IF BACHELOR *u* MALE. *u* , for *u* of type <e>

Here *u* is another unification variable; it is a placeholder for something of type <e>, and can unify with any entity of type <e> in the model. The change of type-face from that used in (8) to (12) is merely to help the reader by distinguishing variables of type <e> from others. The Meaning Postulate thus functions to restrict the meaning of BACHELOR if the denotation of MALE is already known, or to restrict the meanings jointly otherwise. It is necessary that the two instances of *u* are unified with the same entity. We must assume that these unification variables are items of LoT. There will need to be several for each type used, but probably no more than half a dozen.[17] They are also required for example in specifying default syntactic or morphosyntactic rules.

3.6 Combinators and the elimination of variables

Classic CCG has several elegant results to its credit, but by combining it with some Minimalist ingredients, we think that deeper explanations become available. Specifically, we propose that LoT itself contains items which never have phonologically overt NL congeners: combinators. Moreover, despite their apparent invisibility, we shall argue that combinators are also part of the syntactic structure of NLs, and that their use allows us to capture generalisations within and across languages, including LoT, in a new way. In a nutshell, they allow a gain in explanatory adequacy.

Combinators (which we will show in underlined bold type) are higher order functors, which introduce more expressive power into the grammar than is allowed simply by the merge under function application of lexical items that have overt NL counterparts: items representing entities, properties, relations, and logical operators, for example. An immediate

[17] For example, 'EVERY *P Q*, *P u* ⊢ *Q u*' from (25iii) below uses one type <e> unification variable, '*u*', and two type <e, t> unification variables, '*P*' and '*Q*'. If the equivalence of (12a) and (12b) above were stated, three unification variables of type <t> would be needed, for *P*, *Q*, and *R*.

advantage of using combinators in the syntax is that they make possible the elimination of variables, where this is more than a matter of mere notation.

Consider first how, without variables, the meanings that NL expresses with reflexives or bound variable pronouns can be represented in LoT.[18] We assume that these two are treated identically in LoT. Our analyses here are illustrative of the simplicity of a combinatorial account of the expression of meanings for which standard logics use variables; we do not necessarily put them forward as accounts of the NL translations. For these, we take it that the introduction of special items (reflexives) restricting the bound variable to a local antecedent is a product of communicative pressures in NL (there are some NLs not making the distinction, at least in morphology, e.g. Old English, Fijian and Guugu Yimithirr: see Levinson 2000: 334-343, Reuland & Everaert 2010). So consider the LoT equivalent of (14a), assuming that an item EVERY, corresponding to English *every*, is available in LoT. With variables, it can be represented as (14b), where the variable marked in bold binds the selection to which the reflexive relates.

(14) a every cat licked itself

b [EVERY CAT] (λx.[LICK $\boldsymbol{x}$] x)

To express this without variables, we may utilise an argument-reducing combinator, **W**, an operator of type <<e, <e, t>>, <e, t>>, which operates on transitive LICK to deliver the meaning required.[19] The Inference rules relating to **W** are as given in (15), where the use of the turnstile represents the legitimacy of inference:

(15) **W** $P\ u \vdash\ P\ u\ u$ where P is of type <e, <e, t>>, and u is of type <e>;
so **W** has type <<e, <e, t>>, <e, t>>

Using this, the required meaning in LoT corresponding to (14) is (16), which could be paraphrased as 'every cat self-licks':

(16) [EVERY CAT] **W** LICK

In LoT, there is no need for the equivalent of the reflexive *him*/*her*/*itself*.

Now consider a simple use of a bound variable pronoun, as in the interpretation (17b) of the English (17a).

(17) a Every girl [said **she** cried]

b EVERY GIRL . λx [SAY [CRY $\boldsymbol{x}$] x]

The thought underlying (17a) can be represented in a logical language with variables as in (17b). If the phrase in (18a) could be 're-bracketed' so that the merges were as in (18b), then the **W** combinator could be used to give the correct variable-free representation in LoT, with

[18] The classic variable-free treatment of quantifier binding is that of Jacobson (1999). For Jacobson, as for Steedman, the combinators license a derivation, with syntax and semantics in parallel. They are not seen as items of LF or LoT, and indeed the elimination of a syntactic 'level' of LF in the account of NL grammar was a desideratum.

[19] As in Szabolcsi 1992. But we argue in Ch 7 that this is not the combinator used in NL where a reflexive is represented.

[SAY CRY] functioning like LICK in (16). With function application as the only merge rule, this cannot be done (SAY needs a type <t> complement, so CRY of type <e, t> will not do). The combinator having the required 're-bracketing' effect is **B**, with Meaning Postulate as in (19a) (where, given the types, the brackets are redundant, as usual). The more general form is given in (19b).

(18) a [SAY [CRY *u*]]

b * [[SAY CRY] *u*]

c [[**B** SAY CRY] *u*]

(19) a [**B** *f g*] *u* ⊢ *f* [*g u*]

b **B** *f g h* ⊢ *f g h* for **B** of type <<β, t>, <<α, β>, <α, t>>>

f of type <β, t>, *g* of type <α, β>, and *h* of type α.[20]

In (19a), the functors *f* and *g* are combined (with the aid of **B**), before *u* is merged.[21] This means that with SAY for *f* and CRY for *g*, there is a meaningful constituent, [**B** SAY CRY], of type <e, t> like an intransitive verb, to which **W** can apply. The LoT corresponding to the bound variable interpretation of NL (17a) can now be given without any variables, as in (20b). The brackets too are superfluous, if **W** has the type in (15) above.

(20) a Every girl said she cried

b EVERY GIRL **W** [**B** SAY CRY]

Now, we see that the bound variable of the lambda calculus representation is also superfluous.[22]

Instead of the version in (20b), perhaps the human computational system might prefer to use one combinator rather than two, for binding a bound variable pronoun, as in (21a), which

[20] The reader may verify that the types given in (19b) enforce the bracketing (merge order) as in (19a). The types for LoT, and NL, will ultimately come from the lexicon.

[21] We provide an explanation that does not rely on using the lambda calculus to define the meaning of **B**. The Meaning Postulate is an adequate implicit definition, and can be used directly (as, by hypothesis, the inference system over LoT would be). The lambda calculus version of **B** can of course be obtained trivially from (19) by using lambda abstraction. Rules for implicit definitions are discussed in Suppes 1957, Chapter 8 (but note that above, we have expressed our definitions directly as inference rules, rather than using equality and then using elimination rules on this); there is a more general philosophical discussion in Hale and Wright 2001, chapter 5. In relation to combinators, see Curry and Feys 1958/68 chapter 2, or Hindley and Seldin 1986 chapter 2.

[22] This characterisation of bound variables would not be adequate for bound variable pronouns in NL. For example, there will be no means of binding a variable in an adjunct.

uses a single combinator **R**, which may be defined in terms of **W** and **B** as **B W B**.[23] Its direct implicit definition is given in (21b):

(21) a EVERY GIRL **R** SAY CRY

b **R** *f g u* ⊢ *f g u u*

where *f* has type <α, <e, t>>, *g* has type <e, α>, and *u* type <e>

so **R** has type <<*α*, <e, t>, <<e, *α*>, <*α*, t>>> [24]

We assume that LoT might use this rather than **W** and **B** if there were storage or memory problems or sufficient need to speed up inferencing.

We introduce one more combinator now, the 'type lifter' **T**.[25] It seems unlikely that LoT in itself would have any need of the type lifter **T** for expressive purposes, but it might for processing reasons (e.g. speed of reasoning on the basis of information encoded as LoT statements). **T** is defined as in (22):

(22) **T** *f g* ⊢ *g f* where type *f* is <*u*>, type *g* is <*u*, t>, and **T** has type <*u*, <<*u*, t>, t>>.

When the meaning required is represented by *g f*; it would merely make inference more complex. There are nonetheless situations where it might be advantageous to formulate a proposition in one way rather than another. This might be for ease of formulation, for making routine inference, or for compact storage. Here for example are four LoT formulations of the same proposition ('Reynard eats/ate Flopsy'), under Functor First:

(23) a [EAT FLOPSY] REYNARD

b **T** REYNARD [EAT FLOPSY]

c [**B T** REYNARD EAT] FLOPSY

d **T** FLOPSY [**B T** REYNARD EAT]

The first (a) is the most compact; the second (b) might be advantageous if I want to meditate on all I know about Reynard (conjoining further predicates), and the fourth (d), might be useful analogously for Flopsy; the third (c) might be appropriate for thinking about what Reynard ate, and then identifying it as Flopsy. Even if only the simplest form (a) is used in the absence of pressure from communicative desiderata when 'thinking in English', we take it that certain combinators, including **B**, are available *ab initio* in LoT, and that others may be constructed by definition using LoT. The latter might be learned during language acquisition.

---

[23] Jacobson (e.g. Jacobson 1999) uses **z**. We prefer to keep a uniform notation, and use bold underlining and capital letters to distinguish our combinators; we use '**R**' for mnemonic reasons, it being the combinator involved in Raising. Those familiar with combinators will recognise that we have '**B**' for '**B**', and '**T**' for '**T**', but there are some discrepancies, such as our '**R**', which does not correspond to the standard '**R**'.

[24] The version with brackets is [[**R** *f*] *g*] *u* ⊢ [[*f* [*g u*]] *u*]

[25] Also known as '$\mathbf{C}_*$' (Curry and Feys 1958/68).

We will introduce the other combinators in the context of analyses of NL, since it is easier to see what their purpose is. We will argue that in NL, the LoT combinators available cannot always be freely introduced, unless licensed. This constraint reflects a compromise between expressivity on the one hand, and predictability to aid acquisition and parsing, on the other. The free use of **T** for instance would predict that at PF, subject>predicate and predicate>subject should be equally available in any language.[26] It is also relevant that it is licensing that makes a combinator sufficiently predictable not to need a PF realisation.

In all, it seems plausible that the representational system for LoT has the properties of a typed applicative syntax under Functor First, where the vocabulary includes combinators; and this is the proposal that we will assume and explore. Striking confirmation of the ordering predictions for mental representations given by Functor First comes from work by Langus and Nespor 2010, building on earlier work by Goldin-Meadow et al 2008. When hearing subjects are requested to gesture the content of pictures showing actions consistent with an agent and a patient, they produce an SOV order even when their native language (Italian, in this case) is SVO. Langus and Nespor argue that these responses conform to the ordering of the conceptual system, not NL. Assuming that a binding 'noun phrase' has type <<e, t>, t> and the verb has type <e, <e, t>>, the order where both subjects and objects precede the verb is as predicted by Functor First.[27] For sentences with a subject and a complement clause ('C'), speakers of Turkish participating in the same experiment produced the order S V C, even when their native language would give S C V. This is in conformity with the order predicted by Functor First for a merge of a clause of type <t> with a verb of type <t, <e, t>>.

We take it as a working hypothesis that only the 'Regular' combinators are available for NL, and presumably also for LoT. These are defined by elimination rules of the form in (24), where ***X*** is the combinator being defined:

(24) ***X*** $f\, u_1\, u_2 \ldots u_m \vdash \quad f\ S_1\, S_2 \ldots S_j$ where each S_i is a combination of $u_1, u_2, \ldots u_m$.

Of the Regular combinators, Curry and Feys (1958/1968) say that they "… have significance because they represent transformations, so to speak, upon the arguments of a function." They further note that **B**, **I**, **K**, **S**, **W**, **C**, ***Φ***, and ***Ψ*** and are regular combinators (Curry and Feys Volume I p. 161).[28] We will be using the first four of these.

We will further suggest that the combinators are all 'orderly' (i.e. not permutators), in the sense that within the string $S_1 \ldots S_j$, no instance of u_n follows any instance of u_{n+k}, unless it is a duplicate. The combinators listed above are orderly in this sense with the exception of **C**. The combinator **R** as defined in (21b) also falls under the definition of an orderly combinator. But it is plain from its definition that **T**, though regular, is *not* an orderly combinator.[29] Hence under our hypothesis, the representation in (23d) cannot serve to underlie SVO order in NL

---

[26] We will use '>' for PF order, and '≫' for scope order, with 'first'/higher scope on the left.

[27] The reader may note some potential oddities here; SVO order is discussed further in Chapter 5.

[28] The underlining for these combinator names is ours.

[29] **T** is Curry and Feys' **C***. The **T** used in bidirectional CCG (e.g. Steedman 2000a), lifts an item of category X into one of category Y/(Y\X), which is order-preserving on the surface because of the backward slash; but this is a different notion from 'orderly' in the semantics.

(we discuss our alternative in Chapter 3, section 3.3). 'Backward' combinators, such as Steedman's 'backward composition' and 'backward crossed composition' are also ruled out.

3.7 Inference with combinators

We have claimed that LoT, including combinators as lexical items, is fit for inference. We have further claimed that inference should be carried out by pattern matching. In the most general combinatorial system, such a procedure is not necessarily valid. However, provided the combinators are typed, as we suppose, such inferences are indeed valid (see Steedman 2000a Chapter 8, §8.2 for discussion).

We will give an example here of inference carried out in the notation we have suggested. Inference must be conducted by using inference-bearing elements from left to right — that is, given the two premises in (25), we know that the first step must involve either EVERY or GIRL.[30] Since GIRL occurs in both premises, suggesting that it might connect the premises usefully, we do not apply any rule for this, but apply the elimination rule for EVERY in (iii) to Premise 1. CRY and GIRL are of type <e, t>, MAISIE of type <e>, and EVERY of type <<e, t>, <<e, t>, t>>>.

(25) (i) Premise 1: EVERY GIRL **W** [**B** SAY CRY]

(ii) Premise 2: GIRL MAISIE

(iii) Axiom: EVERY *P Q*, *Pu* ⊢ *Qu* (parallel to (8))

Here, EVERY *P Q* unifies with premise 1, so that *P* is necessarily instantiated by GIRL, and *Q* by [**W** [**B** SAY CRY]]. Now *Pu* will unify with premise 2, with *u* instantiated as MAISIE. The inference rule now reads:

(iv) EVERY GIRL **W** [**B** SAY CRY], GIRL MAISIE ⊢ **W** [**B** SAY CRY] MAISIE

At the next step, we use the 'elimination rule' related to **W** (the one without **W** in its consequent), from (15).

(v) inference rule: **W** *P u* ⊢ *P u u*

On unifying the left hand side here with the deduced '**W** [**B** SAY CRY] MAISIE', we obtain

(vi) **W** [**B** SAY CRY] MAISIE ⊢ [**B** SAY CRY] MAISIE MAISIE

Next, the inference rule for **B** is required, given in (vii), and then matching and unification with the initial phrase in the consequent of (vi), to give (viii):

(vii) Elimination rule: [**B** *f g*] *u* ⊢ *f* [*g u*]

(viii) [[**B** SAY CRY] MAISIE] MAISIE ⊢ [SAY [CRY MAISIE]] MAISE

The consequent here is the information deducible from the two premises. Note that the brackets are just for the reader; all the required operations can be performed simply by pattern matching.

[30] This reliably ensures that the type for the whole is <t> — a necessity for the inference rules given.

The inferential steps will be shorter if **R** is used instead of **W** and **B**, here. This might suggest that in LF, and perhaps LoT, **R** is associated with 'bound variable' meanings.

With suitable additional axioms relating to equality, all the turnstiles in the elimination rules may be replaced with equality. Sperber and Wilson 1986/95 argue in relation to *and* that pragmatic processing must eschew introduction rules, so this might be suspect. However, if NL contains lexemes whose LoT is (Functor First) '∧', of type <t, t>, then some human must not only know the appropriate inference rules that define the meaning of this item, but be capable of understanding that a propositions of the form '∧ p q' is valid in situations where the result of applying the inference rules would be valid. Without this, no non-imitative utterance would ever contain *and*. But this is exactly to understand the only correct 'Introduction Rule' for '∧'. The same will apply to any LoT item whose utilisation by the human mind can be established.[31]

4 Natural Language with combinators

4.1 The necessary ingredients of an NL grammar

Under minimalist assumptions the grammar of any NL is required to produce representations adequate to interface with the transducers leading to the input-output (sensory-motor) systems on the one hand, and the cognitive (Conceptual-Intentional) system on the other.[32] That is, they must meet a condition of legibility (Chomsky 2000a: 94f., 2002: 107f.). As argued above, we take the interface to the cognitive system to consist of an LoT representation, whose syntax is a categorial grammar in Polish notation, and where combinators occur in the lexicon of LoT. Meanings are interpreted with respect to the models of the world being considered by the thinker. Any item, whether obtained as a unit (from the lexicon) or constructed by Merge, should have an interpretation at each interface. If I am the speaker, my LoT representation for an NL phrase is unambiguous (though in the case of a pun, I may have two unambiguous LoT representations in mind). But for various reasons things are not as simple for the hearer. The combinators are not overtly distinguished and the PF of an item often underdetermines the LoT item, so that sometimes there may be several legitimate construals of an NL form, with pragmatics being invoked to help choose among them. Even then the interpretation as given by the hearer's LoT need not represent the intended explicature of the utterance, as the hearer may misunderstand it, incorrectly identifying referents of pronouns, for example. If the speaker is sufficiently competent, he will of course take account of the problems a hearer might have, and may adjust the message to facilitate ease of processing and retrieval of the intended effects. Relevance Theory is a theory of hearer competence with respect to comprehension under this assumption. Pickering and

[31] Sperber and Wilson were concerned to eliminate the possibility of ineffective but valid logical moves in human reasoning. For discussion of various ways to constrain patterns of inference suitably, see Uchida 2013 .

[32] This observation is essentially unaffected by recent changes in Minimalism in which PF and LF are no longer levels of representation internal to the grammar (Chomsky, 2008; cf. Richards 2007).

Garrod (2013) offer an integrated account of output and input processing which seems to be compatible with Relevance Theory.

The task of syntax is to mediate between the interfaces with as little apparatus which cannot be seen as interpretable at one or other interface as is possible (Chomsky's 'Interpretability' condition).[33] The computational system must, however, be able to read and utilise at least some uninterpretable (purely syntactic) information associated with the items being merged. In particular, on the assumption for which we have argued, that LoT conforms to a simple Polish notation representation, the syntax of NL could map at PF onto either Functor First or Functor Last (with 'First' now referring to the temporal dimension of speech or sign), and there should be very restricted parametric variation.[34] There could not, for instance, be any variation depending on a noun vs. adjective classification, nor would non-adjacent morphological processes be possible. Manifestly, this is untrue, but let us try to determine what additions are essential.

Consider first the question of linearizing and ordering an NL representation. Since LoT is linearized, LF will be linearized by the same means. If there is a real choice between Functor First and Functor last in the mapping to PF, languages would largely manifest the order 'SOVX', or the reverse, XVOS, for X a clausal argument of the verb. But the reverse is as far as we know unattested as an underlying order, while SOVX is frequent. Further, Dryer 2000 notes that "As Greenberg (1963) observed, conditional clauses exhibit a universal tendency to precede the main clause". This is in line with Functor First. We conjecture that Functor First is universal, and proceed on this basis.[35] There is probably, a cognitive explanation for this fact, if fact it is, though it might simply be a function of how orderings are stored and utilised in the brain, or, as Biberauer et al 2013 argue in relation to Kayne's LCA, a 'third factor' result. But our claim here rests primarily on the beneficial effects of assuming this LoT base for the linguistic description of individual languages, with consequent implications for the minimization of UG. A lack of choice here should also simplify the acquisition process.

In producing a Minimalist grammar of NL, the simplest possible addition to a pre-existing LoT would be to pair each required LoT item in an NL lexicon with some representation of a sound or sign which could be interpreted by the sensory-motor interface, with nothing but the syntax given by LoT types under Functor First. We claim that this is indeed the grounding for NL, but that for a number of reasons, humans have found it too restrictive, and have augmented or subverted it in several ways. The subversions are mainly to be accounted for within Pragmatics, which we trust the reader to supply as necessary. The augmentations will be introduced successively.

---

[33] Of the hypothesis that there is no such apparatus, Chomsky 2000a: 113 says it is 'transparently false'.

[34] There might be some due to the choice of the order of the selections of a head; that is, the equivalent of *know* might be encoded so that it had type <e, <t, t>> or <t, <e, t>> — or perhaps both, for each of two items related by a Meaning Postulate. These might be useful in inference, with one serving much like a passive.

[35] The best places to look for 'Functor temporally Last' languages would be OVS and VOS languages, or any language without quantifiers.

In what follows here, we initially abstract away from all morphological matters, and ignore tense, possible worlds, and intensionality. Given that NL must be learned, the propriety of such abstractions is probably uncontentious.

The minimal requirement on a generative grammar of a natural language is a vocabulary associating conceptual meaning and a mental representation usable for externalisation, and a recursive definition of syntactic well-formedness that supports semantic compositionality. For the representation relating to externalisation, we assume PF (phonological form), but for convenience, give orthography. For the conceptual meaning, we might use a well-formed phrase of LoT, LoTF. This will include any combinators required.[36] However, for expository reasons, we will use the label 'LF' for an LoTF which is associated with a well-formed NL structure. We generally use small capitals to represent LoT, as above, and likewise for LF. For mnemonic reasons, we use OFTEN in LoT and LF to correspond to the NL item realised with PF *often*, and so on, when practicable.

It is not obviously necessary that every minimal LF item correspond to a minimal item of LoT, or vice versa. We consider a few cases where arguably there is no identity of LF and LoTF. For example, if I learn the word *bachelor*, and that its meaning (suppressing combinators) is the conjunction '∧ NOT-MARRIED MAN', I might set up features associating this with the PF /batʃələ/. Later, I might add a new simplex item 'BACHELOR', to my LoT, with a Meaning Postulate associating it with the conjunctive phrase. Whether subsequently a simplex LoT item is established might depend on how often the alternative complex LoT phrase is utilised. Alternatively, as soon as I hear the new PF word *bachelor*, I could immediately set up a new LoTF to relate to the PF word, whose details, such as the logical equivalence to '∧ NOT-MARRIED MAN', could be associated with this new LoT item. There is evidence that this second strategy of 'mutual exclusivity' is that used by young children (Markman 1990, Lust 2006:235), but the behaviour of adults might well not be the same. For instance, an adult might associate a single LoT item CAT (for the domestic cat) with two distinct PF outputs, /mɔgiː/ vs. /kæt/. There are differing intuitions about this mental coding, we do not know of any pertinent experimental evidence, and nothing substantial appears to hang on it, so we will make whatever decision seems best in each instance.

We now turn again to the necessary ingredients of the grammar.

Syntactically well-formed objects vary in size from a word through a phrase to a complex clause. The simplest workable method to obtain a phrase is by iterated binary merge,[37] obtaining one mother item from two daughter items. Here 'mother' is not intended to indicate any commitment to trees as mental objects, but simply as a term to denote the properties of the complex resulting from performing a binary merge on two items, the 'daughters'.[38] In the first instance what is merged will be two items drawn from the lexicon; subsequent merges can combine such complex items (constituents) with each other or with another lexical item.

[36] In CCG as in Steedman 2000a, the combinators are represented only as the licensing condition for well-formed Merge. But the representation of meaning (given in a lambda calculus notation) is obtained recursively at each merge by using the meaning of each combinator.

[37] cf. Chomsky, 2004:108.

[38] See Marcus 2013 for arguments that trees are implausible as mental constructs.

The grammatically relevant information relating to the mother, each daughter, or a new item drawn from the lexicon, is encoded in its 'label'. Items with identical labels should behave identically. To help the reader understand the well-formedness of a structure, we often include the type in the label, though strictly this is not part of the NL syntax, but rather that of LoT.

In the simplest grammars, merge takes place without reference to the internal structure of the items merged. So far, only the type of each of the daughters is accessible, and contributes to the properties of the new item. In current minimalist grammar, Merge, the only structure-building operation, is licensed by the discharge of some feature (Pesetsky and Torrego 2006). In a categorial grammar, we likewise take the basic requirement on Merge to be equivalent to selection feature discharge. In LoT, and concomitantly in LF, one such feature must be the semantic type. The minimal grammar would then produce under Functor First sentences of type <t> like those in (26), using just the basic types <e>, and <t>, and taking proper names to have type <e>.

(26) (a) HUNGRY FIDO (b) BESIDE MARY FIDO (c) BARK FIDO

HUNGRY FIDO *BESIDE MARY FIDO* *BARK FIDO*

Under 'Functor Last', the order would be the mirror image of that given.

Manifestly, the grammar for NL needs augmenting. With just types, both SOME DOG BARK and SOME BARK DOG would be equally well-formed, and have the same meaning. But NL items are not classified just by type; they require syntactic category. The first operand of the quantifier SOME must be a noun phrase, and the second (for a propositional meaning in English), a verb phrase. As is well-known, the same conceptual content can be expressed by two NL items of different NL syntactic categories (*slow*, *slowly*), or with different selection requirements (*likely*, *probable*), or one LF, PF pair showing two or more distinct sets of selection requirements (*impossible*). The categorial system classifies the NL words and phrases for syntactic merge, and is a finer-grained system than that of the types, which classifies LoT concepts for syntactic merge.[39] For example, a propositional attitude verb selecting for the type 'proposition' can in principle be discharged by an item of the category of a binding noun phrase (informally, 'DNP') or a finite clause (informally, 'CP'),[40] and conversely, a pronoun can correspond to the type 'entity' or 'proposition'. A compositionally orientated grammar requires that Merge is specified both for the semantic effect of merging a pair of conceptual items, consistent with the types, and for the categorial effects of merging a corresponding pair of category-labelled items. It follows that selection feature discharge subsumes both semantic (type) feature discharge and categorial feature discharge, which must be compatible with respect to the structure generated. The proper discharging of type selections is in part a stronger version of one half of the Theta Criterion (Chomsky 1981: 36), in that it demands the proper discharge of all selections — those for propositions, as well as those for entities — if a proposition, of type <t>, is to be constructed.

[39] The addition of categories to the syntactic inventory adds apparently meaningless material to the cognitive load, but presumably aids recall, identification and pattern-recognition in the same way that adding colour does in visual perception.

[40] Grimshaw 1979 is the locus classicus for verbs selecting clausal complements.

We are going to assume that the category of an item is given in the lexicon. The idea that there are category-neutral 'roots' that acquire category in context (e.g. Chomsky 1970: 190, Distributed Morphology as in Halle and Marantz (1993)) seems to be inimical to compositional semantics based on LoT, since the 'root' postulated may make no fixed contribution to the interpretation (Harley 2014, Acquaviva and Panagiotidis 2012; see also Borer 2009, 2014). The Distributed Morphology notion of 'root' might better be taken to correspond to our morphosyntactic 'word-name', argued for in detail in the next chapter. But we note that in the overall system as proposed here, there are indeed semantically interpretable but category-neutral items: items of LoT. For example, a predicate of type <e, t> in LoT, has a type but no category; it might in principle in some particular language map onto an NL item with the properties of an adjective or a verb or a noun or more than one of these (e.g. LoT SQUARE as adjective *square* or noun *square* in English, or PRETTY as an adjective in English but a verb, *ge*, in Nupe). The category assigned to an LF item is not random, but nor is it governed by a fixed rule. The defaults are presumably based on commonalities among the inference rules relating to the relevant LoT items — that is, a category like NOUN is like DOG, in that it labels a set of things (words, in this case, rather than animals) that have interesting properties in common.

A categorial grammar uses categories to encode constraints on merge additional to those given by the type. The basis of the Categorial Grammar used here is a parallelism between the rules applying to syntactic category selection features and those applying to semantic type selection features, making it the simplest kind of grammar capable of fulfilling the desiderata mentioned. This parallelism is a prerequisite for the 'Rule to Rule' hypothesis (as in e.g. Bach, 1976:183), whereby the semantic value of the phrase may be computed at each Merge on the basis of the properties of the pair of lexical items or phrases merged. Under a Rule to Rule system, any phrase constructed has an associated meaning. Syntactic and semantic parallelism is not only economical, but aids acquisition. Further, one effect of categorial distinctions is to increase the predictability of the structures licensed by the grammar, which probably aids both parsing and learning. Children are differentially sensitive to their own names by 6 months and to phonetic properties such as stress before a year (Mandel et al. 1995, Shukla et al 2011). Children before the age of two learn to discriminate between relational (lexical/substantive) items and certain 'functional' items, probably on distributional grounds including stress (see e.g. Hochmann & Mehler 2013 and references therein). At the two-word stage, their output consists almost entirely of the former (Radford 1990).

The standard CG notation gives compact and explicit lexical entries, where the category of each required selection is encoded in the category name, just as the type of an item encodes its selection requirements. We start with a set of atomic category labels, and then construct complex category labels from these. The idea is that the category label gives information about the distributional status of an item, whether it be simple or phrasal. For example, a simple transitive verb like *hit*, and a phrase like *pass to Mary*, behave in the same way syntactically: both require two noun-phrase arguments and when provided with these and finite Tense, form a finite clause. Category labels need internal complexity to encode this. A simplified label of the kind we will use much of the time for such items is 'V/D/D'. Here, the 'V' contrasts with say A or P or N, for adjectival, prepositional, or nominal items (heads or phrases), and the two '/D' symbols juxtaposed encode the selections for arguments like *Mary*.

'V', 'D', 'A' and so on are the atomic category labels used either alone or to form complex category labels such as 'A/D'.[41] Note that our use of these complex labels has no directionality encoded in the '/', unlike most Categorial Grammars, since the ordering of two items follows rather from Functor First.

Categorial Grammar automatically accrues certain advantages as a function of its structured categories. One of the more important is that it gives a direct and simple account of adjuncts, which have been a long-standing problem within the P&P tradition. The notion 'specifier' is otiose (Cormack 1999, Chomsky 2010b, 2013). Similarities and differences between 'adjuncts' and 'specifiers' fall out from the CCG specifications obtained on distributional grounds of the items usually ascribed to these classes. Other notions such as unaccusativity are also usefully illumined.

We will explicate all these properties below, but first enter a caveat. A simple categorial grammar is weakly equivalent to a context free phrase structure grammar.[42] Neither is adequate for an explanatory account of NL, so various augmentations are required. We have already argued that LoT needs to include some combinators in its lexicon. We assume then that sufficient combinators are available as items of NL too. That is, unlike standard CCG, we treat the combinators as lexical items in NL (as in LoT), so that the only Merge operation is still function-argument application. With the inclusion of suitable combinators, we will argue that the required syntax and semantics of LF is obtained, including provision for LF-interpretable 'displacement', for *wh*-movement and A-movement, and for some scrambling.

However, we argue that for surface PF, it is necessary to appeal to some further displacement beyond what is licensed in this way by the combinators. For this, in departure from standard CCG, but in line with minimalist approaches treating 'Move' as a side effect of 'Agree', we argue for an analysis entirely in terms of morphosyntactic Agree: that is, the apparent "displacement" of the PF canonically related to a sign, leaving the LF-interpretable part *in situ*. This sort of 'displacement' accounts for 'head movement' (chapter 3) and phrasal movement (chapter 4). To implement this, and also to account for agreement and some phrasal displacement, it is necessary for lexical entries and Merge to encode and manipulate morphosyntactic features as well as categorial features (but not to encode PF directly). The morphosyntactic features produce effects closely related to those of 'Agree' in Minimalism, and likewise require a separate 'Spell-Out' function to produce PF. Until it becomes relevant to consider morphology in any detail, we will refer to this as 'PF displacement'. We claim that nothing of the underlying categorial syntax changes when the PF displacements are added to the overall grammar. We ask the reader to take on trust these further parts of the grammar until Chapters 3 and 5. It is largely in differences in the morphosyntactic domain that the systematic contrasts among different natural languages are accounted for, though combinators may also be differentially licensed in one or another language for certain word-classes.

---

[41] Whether the complex labels for categories could be dispensed with and the selections reformulated in the manner of the morphosyntactic features relating to Agree is discussed in section 10.

[42] See Wood 1993 for discussion and references. See also Pollard (1988: 394) who points out that different notations lead to different sorts of generalisations. For the power of CCGs, see Steedman 2000a § 8.3.

We have now suggested that an item drawn from the lexicon associates a number of features. First, there is the LF, an item of LoT, which is associated with a type. Second, there is its category. Third, there are its morphosyntactic features. In the usual case, lexical items are not phrasal, and we refer to the item (the feature bundle) as a lexeme. For mnemonic purposes, we use an italic form of the LF as the lexeme name where possible. We may now for instance refer to the lexemes *CAT* and *MOGGIE*, both associated with the LoT CAT. These items '*CAT*' and '*MOGGIE*' are also items of LoT used in thinking about the English language (corresponding to one use of the word 'word'). Encyclopaedic information about usage can be associated with these items.

In Chapter 3, we will argue that the minimal morphosyntactic feature for a lexeme consist of a pairing of a word-class and a lexeme name, such as <NOUN: *MOGGIE*>. This pair is the canonic morphosyntactic feature of the lexeme *MOGGIE*. 'NOUN' is itself also an LoT item, being a property of certain lexemes, including *MOGGIE*. It is necessary to know the word-class of *MOGGIE* or *SEE* to know how to form a plural, for example. In contrast, it is the category, N/D, of *MOGGIE* or V/D/D for *SEE*, together with its type, that enters into structure building, Using such features, we can say, for instance, that the notion 'auxiliary' is morphosyntactic, and that AUX *HAVE* has category V/D/D.

4.2 Outline

One of our aims is to provide an account of the relations between NL and LoT where, to a first approximation, for a particular expression of NL (a phrase or a unitary item), this can be reduced to the relations among the three elements:

(27)

LF

MSyn

PF

Here, LF is a well-formed LoT expression, and PF is a phonological form, well-formed according to the phonological rules for the language. LF items are, first, those drawn from that subset of LoT that has been lexicalised in the particular NL in question, plus the permitted combinators, and second, those complex items properly generated by Merge. LF inherits the syntax of LoT, but additionally, satisfies the syntax of the categorial system. The MSyn consist in an ordered array of morphosyntactic feature bundles, derived initially from the lexicon, and subject to Merge along with the associated units of LF. The morphosyntactic merge rules determine the value on the mother at each merge, just as function argument application determines the value of the type, category, and semantic value at the mother at each merge. We take it that the well-formedness of units of these two kinds and their merge rules constitute narrow syntax in the sense of Hauser, Chomsky & Fitch 2002, Chomsky Hauser and Fitch 2005. Because of the underlying LoT system, a well-formed string consisting of <LF, MSyn> pairs will unambiguously represent its own structure. PF is obtained by a mapping, 'Spell Out', from Msyn, but provision has to be made for omitting duplicates formed under Agree. We leave this aside for the moment.

Although empirical evidence is hard to come by, we assume that processing in LoT operates both top-down and bottom-up and, most probably, in parallel. In linguistic

performance in a NL it is likewise plausible to assume a more complex procedure than a purely bottom-up analysis of most competence theories (see for example Pickering and Garrod 2013). In giving our account of how some proposition in LoT may in principle be externalised, however, we idealise away from this complexity and treat thought-formation in LoT purely bottom-up to mirror the compositional bottom-up account of well-formedness conditions in CCG or Minimalism. That is we deploy a competence account of the structure of thoughts to mirror the competence account of NL sentences.

With this simplifying assumption, an idealised outline of the information flow from LoT to PF might be as follows:

1 Formulate the required message in LoT, using only LoT items that have congeners in the chosen output language (which includes some combinators). The message must be well-formed according to the types, with binary Functor First Merge under Function Argument Application. It must also fulfil pragmatic desiderata.

2 From the lexicon, select such a matching NL item to associate with each LoT item so as to respect syntactic category as well as type in the binary Merge.

3 Associate a set of Morpho-Syntactic (MSyn) features with each NL item, according to the choices available in the lexicon. The rules for MSyn are primarily percolation and unification of partially underdetermined features ('Agree'). MSyn features are responsible for agreement (in a broad sense), for 'head-movement', and for some phrasal displacement. MSyn features account for all syntactic well-formedness conditions not covered by the merge rules for types and categories, and are the major source of syntactic variation between languages.

4 Use the fully specified MSyn features and the phrasal structure given by narrow syntax as input to the MS→PF mapping, to obtain the PF representation. The resulting PF will be presented to the output system (for speech or sign).

Of these four groups of procedures, the Merge rules of 2 and 3, together with generalisation determining what kinds of lexical item are licit in a given language, constitute 'Narrow Syntax'. Because MSyn items cannot merge independently of the structure given by LF items, and Spell Out too utilises LF structure for identifying duplicates, we take the grammar to be monostratal.

5 The architecture of a Minimalist Combinatorial Categorial Grammar

5.1 Categories and Merge

We argued above that categories were necessary in NL. In particular, categories are needed for subcategorisation (selection constrained by category). Here, we argue for the particular notation and associated Merge rules of a categorial grammar. The main reason for using this notation is that it is readily related to the type system of LoT, in that it represents the relevant property (type, category) of the arguments selected by a given item, and the property (type, category) of the item resulting after merging these arguments. As in the type system, the

argument order is specified. The notation is also usefully compact, and the well-formedness of merge is probably easier to read off than in the type-system, under Functor First.

Categorial grammars usually choose the smallest possible set of atomic categories. The minimum requirement for a simplistic grammar is for two, usually 'S', of type <t>, to represent the category of a clause, and another, 'NP' of type <e>, to represent the category of a noun phrase. Then, instead of introducing a new category for verbs, an intransitive verb or a predicational verb phrase, whose type is <e, t>, can be given the category 'S/NP' — a category which yields category S when merged with an argument of category NP. Similarly, a transitive verb, of type <e, <e, t>, has category (S/NP)/NP. Notice that the atomic categories, 'S' and 'NP', may and usually do relate to phrasal items, whereas the syntactically simple verb has a complex category label.[43] We will use the categorial system of selection features, but with labels closer to P&P usage than to Chomsky (1965). We start with atomic category labels that are mnemonic for the default word-class of the head of the category. Making the temporary assumption (to which we return) that the verb is the head of the clause, we will have 'V' instead of the standard CCG's 'S'. The category of lexical verb then will be of the form V/X, or (V/X)/Y … where X and Y etc. are the categories of the selections required by this verb.[44] The V here is the 'goal' of the category. Similarly, for the saturated projection of a noun, for which we use 'N'. It should be emphasized that, in contrast to traditional Phrase Structure grammars, 'V' and 'N' do not mean 'verb' and 'noun'. Rather, 'V' is the category of a well-formed syntactic object, typically a phrase — one whose head is typically a verb and which has no (remaining) argument selections: it is the label of a potential mother category in some verbal projection. The notion 'verb' is both syntactic (relating to category), and morphosyntactic (relating to syntactically conditioned morphology; chapter 3); for the latter we use the notation 'VERB', and similarly 'NOUN' for a noun.

Atomic categories contrast with complex categories of the form X/Y, X/Y/Z, etc., where X and Y and Z may be atomic or complex. Since the category has to be learned, there is a default simplicity assumption that the selections made by a head are simple, restricted to the atomic. In fact we will find that there are only three classes of lexical items that require non-atomic selections. These are the combinators, the semantic binders and a small class of arity operators such as passives and applicatives. A similar default assumption holds for types.

A complex category is one that behaves as if it has a selection feature, or an 'edge feature', in the sense of Chomsky 2008: 139. In 'X/Y', Y is the selection feature, which permits the merge of this item with some other item of category Y. In X/Y/Z, merge with an item of category Z is permitted.[45] That is, by convention, for categories as for types, the outer selection is that available to be discharged under function-argument application. On merge, the resulting category will not have a selection for Z, giving the impression that the feature is deleted. What remains ('X/Y', here) is the category of the phrase created by the merge — the mother of the binary branching phrase in tree notation. If the category of the phrase is still complex, as it is here, it licenses another merge, this time with an item with category Y. After

[43] These labels use brackets, because of the infixed slash. An initial slash would permit a bracket-free notation (see Potts 1973, adapting a notation from Ajdukiewicz (1935/1967)).

[44] We will stop using 'category label', and simply use 'category' where no confusion arises.

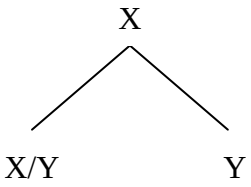
[45] That is 'X/Y/Z' is to be understood as '(X/Y)/Z'.

this second merge, the category of the mother (i.e. of the whole phrase) will be X.[46] We may refer to this final atomic category as the 'goal' category of the complex X/Y/Z; it can be read off the complex category as the initial atomic category. Thus a category labelled X/Y selects for a sister Y, and may merge with such an item to form a phrase of category X.[47] In this merge, the category X/Y can be thought of as a 'functor' (that is, an item which represents a function) from category Y to category X, so that the mother category is obtained by function application.[48] Under the 'Functor First' notation generally used, we have the structures in (28). In (a) we show just the categorial merge. In (b), we show the notation for a phrase, where the categories are shown as subscripts on lexical elements or phrases, *F, G, H,* etc. The tree version is shown in (c).

(28) a X/Y . Y => X

b [$F_{X/Y}$ G_Y] $_X$

c



This merge conforms to a weaker version of the desideratum 'the selector projects' (Chomsky 2000a:133-134), in the sense that the goal category of the mother is that of the selecting head. Here, the category X may either be an atomic category, or it might itself be complex, consisting of say U/V. Putting U/V for X gives us a category (U/V)/Y, which can be merged successively with two arguments, by two applications of (28a), as in (29a) (i) and (29 a) (ii). (29b) is the usual notation we use, where *f, g* and *h* are lexical or phrasal items. (29c) shows the same information in tree form.

(29) a (i) (U/V)/Y Y => U/V ; (ii) U/V V => U

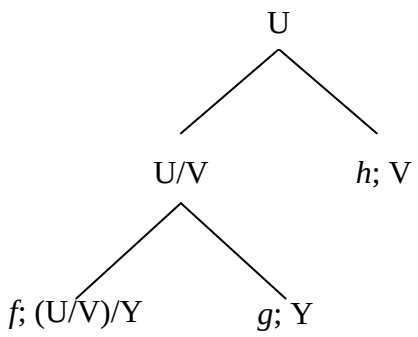
b [[$f_{U/V/Y}$ g_Y]$_{U/V}$ h_V] $_U$

[46] The notation is a bit like X-bar theory in reverse: the category with least annotations in CG is the top of the tree, and the one with most at the bottom, whereas with X-bar theory, the fewest bars are at the leaf, and the maximum at the top. This reversal in effect makes something much like the X-bar-levels of pre-minimalist P&P grammars admissible without violating the Inclusiveness principle (Chomsky 1995:225).

[47] 'X' is to be interpreted as a maximally underspecified category, and likewise for U, V, W.

[48] As discussed in the Preliminaries chapter, function application or some analogue is taken to be provided as an architectural universal by the processing system of the mind/brain.

c



This kind of tree, with a complex left branch, occurs wherever one item *f* (typically, an operator or a binder) selects for two items, both of which are merged on its right in canonical fashion. Further, such a structure can **only** occur when the initial item makes two selections.

Three-argument structures are as easily produced, with one more level of branching on top, from a head whose category is ((U/V)/W)/Z.

It is important to note that assigning a head a category such as ((U/V)/W)/Z gives not only the number (three) and categories (V, W and Z) of the several arguments, together with the category of the saturated phrase resulting from saturating the selections (U), but the order in which the selections can be discharged under successive merges by function application, and the phrases successively constructed.[49] The only way to match up (((U/V)/W)/Z with the X/Y in the rule in (28) is to put ((U/V)/W) = X and Z=Y. Then Z, the outermost, must be discharged first. That is, we are putting forward a theory of the lexicon and merge under which selection order is critical to merge. Syntax is going to depend crucially on this ordering, so its easy learnability is essential (see further in section 8.1). Categories are justified and learned largely on the basis of the gross distribution classes of items with respect to phrases. In contrast, word class, such as VERB, will be justified and learned on the basis of distribution classes of items with respect to morphosyntactic regularities.

It is worth pointing out that the linearization given by Functor First is indifferent to whether the items are heads or phrases. Thus in (28), X/Y might be a head, say a verb, and Y its CP phrasal complement, in line with Kayne's (1994) LCA. But in (29), U/V might be a noun phrase (section 8) or an adjunct (section 7), and V might be a simple verb. The Functor First ordering in these instances is in violation of the LCA, and is not consistent with Chomsky 2013, since here the phrase precedes the head in the merge. Given that languages vary with respect to the PF orders of the corresponding items, no conclusion can be drawn immediately on this basis.

What has been sketched above is rudimentary, and will be expanded below and in later chapters, but it suffices to illustrate the basic syntactic properties of a Categorial Grammar. In particular, the grammar utilising only categories gives syntactic structure, but not necessarily PF or LF. In a pure categorial grammar, PF and LF (or semantics) can be read off the

[49] That is, instead of the head being a function from an ordered set of argument-categories to a goal (mother) category, it has been 'Curried', yielding a function from one argument-category to a function from another argument-category to a goal category.

syntactic structure according to the rules in (30), '>' represents PF sequence, with the item on the left as the first.

(30)

	functor	argument		result (mother)	
category:	X/Y	Y	→	X	(function application)
type	<α, β>	<α>	→	<β>	(function application)
semantics	*f*	*g*	→	*f g*	(function application)
PF	*F*	*G*	→	*F > G*	(sequence formation)

Notice that the type and category notations are in mirror order, with the mother final in the category but initial in the type. As noted above, in the highly constrained Categorial Grammar we posit, PF cannot be so simply given, but requires the intermediary MorphoSyntactic system.

In CCG, as in formal semantics standardly, discharged selections leave no imprint in the mother category.[50] In CCG, the absence of any record of discharged features embodies the claim that a phrasal item that has acquired a certain category after discharge of some selections, and an item with the same category but drawn directly from the lexicon, behave syntactically in exactly the same way. For example, a transitive verb merged with its direct object, and an intransitive verb drawn from the lexicon both have the same category, and will behave alike syntactically, if we abstract away from morphosyntax. Taking the CCG category to be the 'label' would satisfy Hornstein's (2009: 59) requirement that the label of a phrase is the same as the label of some non-phrasal item, despite the fact that they are complex. Further, if we (incorrectly) think of the various ingredients of a category as 'features', then in a merge as in (30), the item with category X/Y has a selection feature that is satisfied in the merge, and it is a feature of this head that projects, as is suggested by Adger 2003: 91 (where say V[*u*X] plays the role of V/X in CCG).[51]

In the next sections, we will discuss the category to be allotted to various sorts of items. We will use the term 'lexical' head to cover both the classes usually called 'lexical' and 'functional', where the term is to cover items from the NL lexicon other than NL phrases (e.g.

---

[50] If selections are considered like theta grid items, then in some notations (following Higginbotham 1995) a discharged theta grid item is projected to the mother category, annotated with a '*' to indicate discharge, so that we might expect 'β*' under X. Some variants of minimalism require discharged (checked) uninterpretable features to be retained in a certain domain, a 'phase' (Chomsky 2001) (see Chapter 3 for discussion in relation to our Agree features), but in CCG, the type and category labels are not decomposed into features.

[51] Since Collins 2002 selection for the category of the sister (e.g. for VP or NP) has been eschewed within minimalism. Categorial selection is mentioned, but often ignored, and the notation where it is used is not uniform. Adjuncts generally remain problematic. Cecchetto and Donati 2010, Donati and Cecchetto 2011 argue for category as the 'label', a feature of the probe, which presupposes categorial selection. See also Georgi and Müller 2010, Stabler 1997.

idioms). The previous 'lexical' heads canonically state relations between entities, so we call them the 'relational' heads. 'Functional' heads include operators (simple adjuncts and the heads of complex adjuncts) and binders (for example, the heads of quantified noun-phrases). The remaining lexical items are the arity-changing heads, such as Passive, and the combinators. Each class can be characterised by its LoT type, though as we will see, that classification allows for some overlap. We begin with relational heads.

(*to be continued*)

References

Abell, F., Francesca Happé & Uta Frith (2000) "Do triangles play tricks? Attribution of mental states to animated shapes in normal and abnormal development". *Cognitive Development* 15:1-16.

Acquaviva, Paolo and Phoevos Panagiotidis (2012) "Lexical decomposition meets conceptual atomism". *Lingue e Linguaggio*, 11 (2): 165-180.

Adger, David (2003) *Core Syntax: A Minimalist Approach.* Oxford, Oxford University Press.

Ajdukiewicz, Kazimierz (1935) "Die syntaktische Konnexität". In Storrs McCall (ed) *Polish Logic 1920-1939*; Oxford, Oxford University Press; pp.207-231.

Bach, Emmon (l976) "An Extension of Classical Transformational Grammar". In *Problems of Linguistic Metatheory* (Proceedings of the l976 Conference). Pp. 183-224. Michigan State University.

Biberauer, Theresa, Ian Roberts & Michelle Sheehan (2013) "On the Mafioso Effect in Grammar". Paper presented at the Biolinguistics Workshop, Lund.

Borer, Hagit (2009) "Roots and categories" Ms.

Borer, Hagit (2014) "Wherefore roots?" *Theoretical Linguistics*, 2014; 40(3/4): 343 – 359.

Carey, Susan (2011) "The Origin of Concepts: A précis". *Behavioral and Brain Sciences*, 34, 113-167.

Carruthers, Peter (1996) *Language, Thought and Consciousness: An Essay in Philosophical Psychology*. Cambridge, Cambridge University Press.

Cecchetto and Donati (2010) "On labelling: principle C and head movement" *Syntax* 13: 241-278.

Champollion, Lucas (2011) "Quantification and negation in event semantics". The Baltic International Yearbook of Cognition, Logic and Communication Volume 6: *Formal Semantics and Pragmatics: Discourse, Context, and Models;* pp.1-23.

Chomsky, Noam (1965) *Aspects of the Theory of Syntax*. Cambridge, MA, MIT Press.

Chomsky, Noam (1970) "Remarks on nominalization." In R. Jacobs & P. Rosenbaum (eds.), *Readings in English Transformational Grammar*. Waltham, MA, Ginn & Co. 184–221.

Chomsky, Noam (1977) "On WH-movement." In Peter Culicover, Thomas Wasow & Adrian Akmajian (eds.), *Formal Syntax*. New York, Academic Press, 71–132.

Chomsky, Noam (1981) *Lectures on Government and Binding.* Dordrecht, Foris.

Chomsky, Noam (1995a) *The Minimalist Program*. Cambridge, MA, MIT Press.

Chomsky, Noam (2000a) "Minimalist Inquiries: The framework". In Roger Martin, David Michaels & Juan Uriagereka (eds) *Step by Step.* Cambridge MA, MIT Press; pp.89-155.

Chomsky, Noam (2004) "Beyond explanatory adequacy". In Adriana Belletti (ed) *Structures and Beyond: The Cartography of Syntactic Structures*, vol. 3; Oxford, Oxford University Press; pp. 104-131.

Chomsky, Noam (2008) “On phases”. In Robert Freidin, Carlos Otero & Maria-Luisa Zubizarreta (eds) *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud.* Cambridge MA; MIT Press; pp. 136-166.

Chomsky, Noam (2010a) “Some simple evo-devo theses: how true might they be for language?” In Richard Larson, Viviane Déprez & Hiroko Yamakido (eds) *The Evolution of Human Language: Biolinguistic Perspectives*; Cambridge, Cambridge University Press; pp. 45-62.

Chomsky, Noam (2010b) “Restricting stipulations: consequences and challenges”. Talk given at the University of Stuttgart, March 24. (The Fromm lecture). http://biolingblog.blogspot.co.uk/2010/05/chomsky-in-stuttgart.html

Chomsky, Noam (2013) “Problems of projection”. *Lingua.* 130:33-49.

Chomsky, Noam, Marc Hauser & Tecumseh Fitch (2005) “The minimalist program”. Unpublished appendix to Fitch et al, 2005. http://www.wjh.harvard.edu/~mnkylab/publications/recent/EvolAppendix.pdf

Cormack, Annabel (1999) “Without Specifiers”. In David Adger, Susan Pintzuk, Bernadette Plunkett & George Tsoulas (eds) *Specifiers: Minimalist Approaches*. Oxford, Oxford University Press; pp. 46-68.

Cormack, Annabel & Neil Smith (2005b) “Linearisation: Adjuncts and arguments”. *University College London Working Papers in Linguistics* 17:111-129.

Culbertson, Jennifer, Paul Smolensky & Géraldine Legendre (2012) “Learning biases predict a word order universal”. *Cognition* 122: 306-329.

Curry, Haskell & Robert Feys (1958/1968) *Combinatory Logic.* Amsterdam, North Holland.

Donati, C., & Cecchetto, C. (2011). “Relabeling Heads. A Unified Account for Relativization Structures.” *Linguistic inquiry*, 42: 519- 560.

Dowty, David, Robert Wall & Stanley Peters (1981) *Introduction to Montague semantics.* Dordrecht, Reidel.

Dryer, Matthew (2000) “Word order” ms..

Fodor, Jerry A. (1975) *The Language of Thought*. New York: Crowell.

Fodor, Jerry A. (2008) *LOT 2: The Language of Thought Revisited: The Language of Thought Revisited*. Oxford, Oxford University Press.

Gergely, György, Z Nadasdy, Gergely Csibra & Szilvia Biro (1995) “Taking the intentional stance at 12 months of age”. *Cognition* 56:165–93.

Georgi, Doreen and Gereon Müller (2010) “Noun-phrase structure by reprojection” *Syntax* 13: 1-36.

Goldin-Meadow, Susan, Wing Chee So, Aslı Özyürek and Carolyn Mylander (2008) “The natural order of events: How speakers of different languages represent events nonverbally.” *Proceedings of the National Academy of Sciences* 105: 9163–9168.

Greenberg, Joseph H. (1963) **“**Some universals of grammar with particular reference to the order of meaningful elements” In: Joseph H. Greenberg (ed.). 1963. *Universals of Language*. London: MIT Press, pp.73-113.

Grice, Paul (1975) “Logic and conversation”. In Peter Cole & Jerry Morgan (eds) *Syntax and Semantics 3: Speech Acts*. Academic Press, New York; pp. 41-58.

Grice, Paul (1982) “Meaning revisited”. In Neil Smith (ed.) *Mutual Knowledge;* London, Academic Press; pp.223-243.

Grimshaw, Jane (1979) “Complement selection and the lexicon”. *Linguistic Inquiry* 10:279-326.

Grimshaw, Jane (1990) *Argument Structure.* Cambridge MA, MIT Press.

Guttenplan, Samuel (1986) *The Languages of Logic: An Introduction to Formal Logic.* Oxford, Blackwell.

Hale, Bob & Crispin Wright (2001) *The Reason's Proper Study: Essays Towards a Neo-Fregean Philosophy of Mathematics.* Oxford, Oxford University Press.
Halle, Morris & Alec Marantz (1993) "Distributed morphology and the pieces of inflection". In Ken Hale & Jay Keyser (eds) *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*. Cambridge MA, MIT Press; pp.111-176.
Harley, Heidi (2014) "On the identity of roots" *Theoretical Linguistics* 2014; 40(3/4): 225 – 276.
Hauser, Marc, Noam Chomsky & Tecumseh Fitch (2002) "The faculty of language: What is it, who has it, and how did it evolve?" *Science* 298 (5598), 1569-1579.
Heim, Irene & Angelika Kratzer (1998) *Semantics in Generative Grammar*. Oxford, Blackwell.
Hindley, Roger and J.P. Seldin (1986/2008) *Lambda-calculus and Combinators: An Introduction.* Cambridge, Cambridge University Press.
Hinzen, Wolfram (2013) "Narrow syntax and the language of thought". *Philosophical Psychology*, 26: 1-23.
Hochmann, Jean-Rémy & Jacques Mehler (2013) "Recent findings about language acquisition". In Massimo Piattelli-Palmarini & Robert Berwick (eds) *Rich Languages from Poor Inputs*. Oxford, Oxford University Press; pp.107-114.
Hornstein, Norbert (2009) *A Theory of Syntax.* Cambridge; Cambridge University Press.
Jacobson, Pauline (1999) "Towards a variable-free semantics". *Linguistics and Philosophy* 22: 117-184.
Jozet-Alves, Christelle, M. Bertin & N. Clayton (2013) "Evidence of episodic-like memory in cuttlefish". *Current Biology* 2013 Dec 2;23(23):R1033-5.
Kayne, Richard (1994) *The Antisymmetry of Syntax*. Cambridge, MA, MIT Press.
Langus, Alan & Marina Nespor (2010) "Cognitive systems struggling for word order". *Cognitive Psychology* 60:291-318.
Levinson, Stephen (2000) *Presumptive Meanings*. Cambridge, MA. MIT Press.
Łukasiewicz, Jan (1957) *Aristotle's Syllogistic from the Standpoint of Modern Formal Logic*. Oxford, The Clarendon Press.
Lust, Barbara (2006) *Child Language: Acquisition and Growth.* Cambridge, Cambridge University Press.
Mandel, Denise, Peter Jusczyk and David Pisoni (1995) "Infants' Recognition of the Sound Patterns of Their Own Names". *Psychological Science* 6: 314-317.
Marcus, Gary (2013) "Evolution, memory, and the nature of syntactic representation". In Johan Bolhuis and Martin Everaert (eds) *Birdsong, Speech and Language: Exploring the Evolution of Mind and Brain;* Cambridge MA, MIT Press; pp. 27-44.
Markman, Ellen (1990) "Constraints children place on word meanings". *Cognitive Science* 14:57-77. Reprinted in Paul Bloom (ed) (1994) *Language Acquisition: Core Readings*; Cambridge MA, MIT Press; pp. 154-173.
Moens, Marc and Mark Steedman (1988) "Temporal ontology and temporal reference" *Journal of Computational Linguistics* 14: 15-28.
Pesetsky, David, and Esther Torrego (2006) "Probes, goals and syntactic categories." In *Proceedings of the Seventh Tokyo Conference on Psycholinguistics. Tokyo*, pp. 25-60.
Pickering, Martin and Simon Garrod (2013) "An integrated theory of language production and comprehension". *Behavioral and Brain Sciences* 36 (4) 329-347.
Pollard, Carl (1988) "Categorial Grammar and Phrase Structure Grammar: An excursion on the syntax-semantics frontier". In Richard Oehrle, Emmon Bach and Deirdre Wheeler (eds) *Categorial Grammars and Natural Language Structures*. pp.391–415. Dordrecht: Reidel.

Potts, Timothy (1973*)* “Fregean Categorial Grammar”. In Radu Bogdan & Ilkka Niiniluoto (eds), *Logic, Language, and Probability*. Boston, D. Reidel.
Radford, Andrew (1990) *Syntactic Theory and the Acquisition of English syntax : The Nature of Early Child Grammars of English*. Oxford, Blackwell.
Rappaport, Malka & Beth Levin (1988) ``What to Do with Theta-Roles", in Wendy Wilkins (ed) *Syntax and Semantics 21: Thematic Relations*, Academic Press, New York, pp. 7-36.
Reuland, Eric & Martin Everaert (2010) “Reaction to: The Myth of Language Universals and cognitive science - Evans and Levinson's cabinet of curiosities: Should we pay the fee?” *Lingua* 120: 2713-2716.
Richards, Marc (2007) “Deriving the edge: what’s in a phase?” MS University of Leipzig.
Rizzi, Luigi (2013a) “Introduction: Core computational principles in Natural Language syntax.” *Lingua* 130:1-13.
Shaumyan, Sebastian (1977) *Applicational Grammar as a Semantic Theory of Natural Language.* Chicago; University of Chicago Press.
Shukla, Mohinish, Katherine White & Richard Aslin (2011) “Prosody guides the rapid mapping of auditory word forms onto visual objects in 6-mo-old infants”. *PNAS* 108 (15): 6038-6043.
Smith, Neil (1983) *Speculative Linguistics*. (An inaugural lecture delivered at UCL, published by the College).
Smith, Neil (2001) “Babes and sucklings”. *Glot International* 5: 13-15.
Spelke, Elizabeth (2004) “Core knowledge”. In N. Kanwisher & J. Duncan (Eds.), *Attention and performance, vol. 20: Functional neuroimaging of visual cognition*. Oxford: Oxford University Press; pp.
Spelke, Elizabeth & Katherine Kinzler (2007) “Core knowledge”. *Developmental Science* 10: 89-96.
Sperber, Dan & Deirdre Wilson (1986/95) *Relevance: Communication and Cognition*. Oxford, Blackwell.
Stabler Edward P. (2010) “Computational perspectives on minimalism”. In C. Boeckx (Ed.) *Oxford Handbook of Linguistic Minimalism*, pp.616-641.
Steedman, Mark (2000a) *The Syntactic Process*. Cambridge, Mass.: MIT Press.
Steedman, Mark (2000b) “Information Structure and the Syntax-Phonology Interface” Linguistic Inquiry, 31.4, 649-689.
Steedman, Mark (2002) “Plans, affordances and combinatory grammar”. Linguistics and Philosophy 25:723-753.
Suppes, Patrick (1957) *Introduction to Logic.* New York; van Nostrand Reinhold.
Szabolcsi, Anna (1992) “Combinatory Grammar and projection from the lexicon”. In Ivan Sag and Anna Szabolcsi (eds) *Lexical Matters*. Stanford CA; Stanford University Press. pp.241-268.
Uchida, Hiroyuki (2013) “Logic in pragmatics”. *Lingua* 133:336-359.
Wood, Mary McGee (1993) *Categorial Grammar,* London, Routledge.