Three ways to link Merge with hierarchical concept-combination

Chris Thornton
Centre for Research in Cognitive Science
University of Sussex
Brighton
BN1 9QJ
UK
c.thornton@sussex.ac.uk

April 8, 2016

Abstract

In the Minimalist Program, language competence is seen to stem from a fundamental ability to construct hierarchical structure, an operation dubbed 'Merge'. A difficulty with this is the hierarchical combinability of concepts. Hierarchical concept-combination is a function the conceptual system possesses inherently. Does this mean the mind has two independent systems for hierarchical construction? Or does it make more sense to think in terms of a unified mechanism? This paper uses a formal notation for hierarchical concept combination to investigate the issue. It shows there are three ways linguistic Merge might be functionally related to hierarchical concept-combination. It also examines the architectural implications that arise in each case.

Keywords: Merge, Minimalist Program, hierarchical concept combination

1 Introduction

Hauser et al. (2002) note that underlying language there must be a faculty that is 'hierarchical, generative, recursive, and virtually limitless with respect to its scope of expression' (Hauser et al., 2002, p.1569). By implication, the language system must have, at its heart, an operator which can relate multiple objects to a single object in the formation of a hierarchical unit. Recursive application of this operator must be what gives rise to structurally complex expressions. The Minimalist Program dubs this fundamental operator 'Merge'. Chomsky deduces its existence as follows:

> An elementary fact about the language faculty is that it is a system of discrete infinity. Any such system is based on a primitive operation that takes n objects already constructed, and constructs from them a new object: in the simplest case, the set of these n objects. Call that operation Merge. Either Merge or some equivalent is a minimal requirement. With Merge available, we instantly have an unbounded system of hierarchically structured expressions. (Chomsky, 2005, p. 11-12)

One problem with this is the hierarchical combinability of concepts. Hierarchical concept-combination is a natural process of thought exercised by us all on a regular basis (e.g., conceptualizing a residence as an entity encompassing a house and garden). The conceptual system must have use of a Merge operator of the type Chomsky describes. Does this imply that there are two such operators, one in the conceptual system and another in the language system? Or is it more plausible to think in terms of a unified entity? Does linguistic Merge give rise to hierarchical concept combination in some way, or is it the other way around?

The present paper investigates this issue using a formal notation for hierarchical concept-combination (HCC). The aim is to provide a precise account of the ways in which linguistic Merge might be connected to HCC. The section immediately to follow (Section 2) examines the hierarchical combinability of concepts from the computational point of view, and introduces the notation used throughout the rest of the paper. The kinds of conceptual representation that can be built in this way are examined, and note is taken of the way they remain semantically precise regardless of complexity. Section 3 then shows that the relationship between Merge and HCC might take one of three forms. The simplest possibility is a completely disjoint relationship, i.e., full independence. But if we assume the mind avoids duplication of effort, a functional dependency of some kind is more plausible. This might either take the form of a 'syntax-first' arrangement in which HCC stems from Merge, or a 'semantics-first' arrangement in which Merge stems from HCC.

Some technical points need to be mentioned at the outset. Material dealing with conceptualization follows the approach of Murphy (2002) in avoiding use of special fonts in the naming of concepts. Where there is any ambiguity, the phrase 'the concept of X' is used to indicate that X is a concept name. The dot '.' is used as a connective in phrasal concept names. The concept of an old man thus has the name 'old.man.' Some of the material dealing with language makes use of interlinear glosses as a way of characterizing syntactic structure. All the glosses presented can also be found in the online resource 'The World Atlas of Language Structures' (Dryer and Haspelmath, 2011). References to the present contents of this resource (as at December 2015) use the acronym 'WALS'. The internet location of the resource is `http://wals.info`.

2 Hierarchical concept combination

Evidence for the hierarchical combinability of concepts is directly available to us all. Hierarchical concept-combination is a conceptual operation that mediates much human thought.[1] It is the process of treating one concept as encompassing one or more others. For example: given the concept of a house, the concept of a garden and the concept of a residence, a hierarchical combination can be formed based on the latter. We can form the idea of a residence *encompassing* a house and garden. Such constructions are readily given multiple levels of organization. Given the concept of a lawn and the concept of a flower-bed, we can form the idea of a garden encompassing a lawn and flower-bed. The two constructions can then be put together to yield a hierarchical combination of two levels. This expresses a still more specific concept—that of a residence constituted of a house and a garden, in which the garden encompasses a lawn and flower-bed. Hierarchical concept-combination is naturally recursive in this way.

A kinship with linguistic Merge is then apparent. This is the recursive operator that is believed to form the functional heart of the language system. Merge is assumed to be the means by which the system builds arbitrarily deep hierarchical structures expressing arbitrarily complex meanings (Chomsky, 1995, 2005). Hierarchical concept-combination fulfils the same function for the conceptual system. This is also a recursive operator, and also the means of building arbitrarily deep hierarchical structures with arbitrarily complex meanings. Should we conclude that the mind deploys two such operators, one in the conceptual system and one in the linguistic system? Or is it more reasonable to assume the existence of an integrated mechanism?

For purposes of exploring the issue, a way of notating hierarchical concept-combination is required. There is no standard formalism for this, and devising one is not without difficulties. A construction of this type is formed by setting one concept into the *encompassing* role with respect to one or more others. The concept in question then fulfils the role of constructive operator; but the arguments to which it applies are also concepts. All the constituents of a construction are thus concepts. The usual approach of allocating different symbols to denote distinct constructive operators (e.g. Sowa, 1984) is ruled out. The notation has to allow *any* concept to be so designated. A second problem is the fact that the encompassed concepts are undifferentiated in role, and arbitrarily numerous.

On the face of it, a predicate-style notation might seem to provide what is needed. Consider the hierarchical combination in which concept X is specified as encompassing concepts Y and Z. (The idea of a residence encompassing a house and garden offers an illustration.) As a way of notating this, we might consider an approach which treats X as a function or predicate, thus:

[1]Hierarchical concept combination is not to be confused with *generic* concept combination. This is a more diverse process that has been modeled in a range of ways (e.g. Hampton, 1991; Thagard, 1997; Rips, 1995; Wisniewski, 1997; Costello and Keane, 2001; Hampton, 1997, 2011).

X(Y, Z)

This is unsatisfactory for several reasons. One problem is that it makes the construction indistinguishable from an expression in predicate logic. This would have a different meaning—the value would be a boolean, rather than a specialization of X. Equally problematic is the way the arguments are presented. The construction implies that Y and Z play distinct roles, which is not the case. The approach is also less than ideal in the way it accommodates embedding. Where constituents are constructions in their own right, it becomes difficult to decipher the hierarchical structure. If one of the constituents of the above construction is a Z encompassing a B and D, the notation becomes

X(Y, Z(B, D))

The imperfect alignment of the bracketing and the hierarchical structure then poses a problem. Using this style of notation, constructions become more difficult to decipher as they increase in depth.

In view of these issues, a square-bracketed, prefix notation is adopted here. A hierarchical combination is specified by placing *all* the combined concepts within square brackets, with the encompassing concept placed first. The concept of an X encompassing a Y and a Z is thus notated by

[**X** Y Z]

The encompassing concept appears as the first element; it is also set in bold for emphasis. The encompassed concepts are the remaining, non-initial elements, and these are not in any particular order. They form a set. In all cases, therefore, it is true by definition that

[**X** Y Z] = [**X** Z Y]

Using Backus-Naur format (BNF),[2] the syntax of a construction in this notations is defined as follows:

⟨spec⟩ ::= ⟨concept-name⟩
⟨spec⟩ ::= "[" ⟨spec⟩ $\langle\text{spec}\rangle^{+}$ "]"

The non-terminal ⟨spec⟩ denotes the specification of a concept. This is defined to be either the name of a concept, or a square-bracketed construction containing two or more ⟨spec⟩. The meaning of a concept name is then the concept that is named, but the meaning of a construction is defined in terms of the 'encompasses' relation. It is the first specified concept taken as encompassing all the others.

Given this way of writing constructions, combinations of more than one level are readily dealt with by embedding. The concept of an X constituted of a Y and a Z, where the Z is itself constituted of a B and a D, has the specification

[**X** Y [**Z** B D]]

[2]A '+' superscript is used to denote 1 or more occurrences of the item.

With the bracketing and hierarchical structure perfectly aligned, the latter becomes more evident.

In general character, this notation is a mix of a prefix system and a programming language.[3] The construction [**X** Y Z] is analogous to (+ 2 3), i.e., 2 + 3 with the plus moved to prefix position, and brackets appended. But whereas the value of (+ 2 3) is a number, the value of [**X** Y Z] is a concept—the concept of an X encompassing a Y and Z. Viewed as a kind of programming language, the notation has a kinship with LISP, a functional programming language often used in AI (McCarthy et al., 1985/1962). It can be seen as a version of LISP in which objects resolve to concepts, and expressions evaluate to structured concepts.

The notation allows a hierarchical combination of any depth to be expressed, and deals with the encompassed elements in a way that emphasizes their identity as a set. Using it, we can begin to explore the diverse constructions that can be put together in this way. Imagine we are provided with a 'kit' of four concepts: the concept of a flight, the concept of a drive, the concept of a journey and the concept of a pilgrimage. The set of given concepts is then: flight, drive, journey, pilgrimage.[4] On this basis, a possible hierarchical combination is

[**journey** flight drive]

This places the journey concept into the encompassing role, with flight and drive as encompassed elements. It realizes the idea of a journey encompassing a flight and drive. The construction replicates the familiar form of a part/whole specification. Another possibility is

[**pilgrimage** flight drive]

This expresses the subtly different concept of a pilgrimage encompassing a flight and a drive. Although the same encompassed concepts are referenced, the construction has a different encompass*ing* concept. It realizes a different idea in result—the idea of a particular type of pilgrimage.

Less obviously, there is the potential for hierarchical combinations which use a single encompassed concept; e.g.,

[**journey** drive]

This constructs the idea of a journey constituted *solely* of a drive. Another combination yields the idea of a journey constituted solely of a flight:

[**journey** flight]

Constructions like these are termed 'singles' and are naturally seen as classifications. The second, for example, can be seen to express the idea of a flight that is, in addition, classified as a journey. It is also worth noting that singles are inherently reversible. An X that is classified as a Y can also be seen as a Y classified as an X. By definition, therefore

[3] It can also be described as a Polish notation.

[4] In all cases, named concepts are assumed to have the implied semantics; e.g., the drive concept is assumed to have the usual meaning of 'drive'.

[**X** Y] = [**Y** X]

Some hierarchical combinations seem nonsensical. Consider, for example,

[**flight** drive]

This expresses the concept of a flight encompassing a drive. In normal circumstances, this would be semantically incoherent. In a world with flying cars the construction might make sense, however.

Given a relational encompassing concept, combinations with more than one encompassed element can express relational schemata. An illustrative example is

[**understanding** teacher lawyer]

This constructs the concept of an understanding encompassing a teacher and a lawyer—an understanding *between* a teacher and a lawyer, in other words. The encompassing concept is implicitly an imposed relation, and what is constructed is a schema in result. But notice that no schema-making apparatus is involved. The concept of understanding provides all that is required. Deployed in the encompassing role, it provides the 'glue' that holds the two constituents together. The relational arrangement is captured purely by hierarchical combination—by treating one concept as encompassing the other two.

Singles can be used to refine concepts of this type. For example, the following two-level combination might be formed:

[**understanding** [**agent** teacher] [**recipient** lawyer]]

This assembles the concept of an understanding encompassing a teacher and a lawyer, in which the teacher is classified as agent, and the lawyer is classified as recipient. It builds the concept of a teacher understanding a lawyer, rather than vice versa.

Does this construction have this exact meaning? It is natural to ask how a notation with no conventional operators can give rise to expressions that are so semantically precise. Why should we view this construction as anything more than a bracketing of concept names? Key to the explanation is the way concepts play different roles in hierarchical combination. Every given concept has the potential to be deployed as the encompassing element in a construction. Every given concept *is* a pre-defined operator in this sense. Given we have precise meanings for all named concepts, we then have a precise meaning for any hierarchical construction formed out of them. This remains the case regardless of the number of levels. The meaning is expressed by stating the hierarchical relations in the specified order. In this case, the meaning is that of an understanding encompassing a teacher and a lawyer, in which the teacher is classified as agent, and the lawyer as recipient—a teacher understanding a lawyer, in other words. A hierarchical combination of any depth is guaranteed to have a precise meaning in this way.

Making further use of structural embedding, we can get closer to the kinds of meaning normally expressed using language. Consider the following, for example:

[**seeing.action** [**subject** John] [**object** [**definite.thing** book]]]

This constructs the idea of a seeing.action encompassing John classified as subject, and a book classified as object, in which the book is also classified as a definite.thing. What is constructed is the idea of some individual John[5] seeing a definite book. This is something that we would express in English by saying *John sees the book*.

A slightly more complex case is

[[**past.behavior** reading.action] [**subject** John] [**definite.thing** letter]]

This is a similar construction except that here the object is a letter, the action is reading rather than seeing, and this is itself classified as past.behavior. This has the effect of placing the reading action into the past realizing a meaning that we would express in English by saying *John read the letter*. This illustrates the way the meaning of a tense can be captured.

A still more elaborate example is

```
[yesterday.event
   [giving.action
      [subject [indefinite.thing man]]
      [object bread]
      [indirect.object John] ] ]
```

Key to the meaning of this is the first encompassed concept. Itself a structure, this expresses the idea of a giving action encompassing an indefinite man (classified as subject), bread (classified as object) and John classified as an indirect object. What this expresses is the idea of an event in which an indefinite man gives bread to John. This event is then itself classified as a 'yesterday.event', i.e., an event occurring yesterday. The final product is the idea of a man giving bread to John yesterday, a meaning we could express in English by saying *Yesterday, a man gave bread to John.*

Some special forms of meaning, such as questions, can also be captured. Consider this, for example:

```
[question
   [[event drinking.action focal.thing]
     [subject [definite.thing teacher]]
     [object [definite.thing [substance water]] ] ] ]
```

The central concept here is

[**event** drinking.action focal.thing]

This expresses the idea of an event encompassing a drinking.action and a fo-

[5]For present purposes, names of individuals are taken to name the concept of the individual in question.

cal.thing. Encompassed by this are water and a teacher, with these being classified as subject and object respectively (and also as definite objects). This idea is then itself classified as a question. The final result is thus (the idea of) a question that asks whether a definite teacher is drinking some definite water. This is something we would express in English by asking *Is the teacher drinking the water?*

These latter examples give a sense of the semantic range that HCC has when applied recursively. They show how the structured meanings we express using language can sometimes by captured purely in this way. They also highlight the operator's unboundedness. In principle, there is no limit on the number of recursive iterations that can be applied. Consequently, there is no limit on the number of hierarchical levels that can be assembled. Hierarchical concept combinations can be of unlimited depth, and can express meanings of unlimited complexity.

3 Merge

The focus can now return to Merge, and the question of how this operator relates to HCC. Signs of a close connection have already been noted. Merge is seen to be the reason language is 'hierarchical, generative, recursive, and virtually limitless with respect to its scope of expression' (cf. Hauser et al., 2002, p.1569). Hierarchical concept combination endows conceptualization with exactly the same properties. Yet the match is not quite perfect. Informally, Chomsky characterizes the behavior of Merge as 'Take two objects, make another object' (Chomsky in: Boeckx, 2009, p. 52). More precisely, the operator 'takes two syntactic objects α and β and forms the new object $\gamma = \{\alpha, \beta\}$.' (Chomsky, 2001, p. 3).

Two issues then arise. In building a hierarchical unit, Merge applies a particular constructive operation. The hierarchical unit constructed from α and β is defined to be $\{\alpha, \beta\}$. It is the set comprising the two constituents. The encompassing concept for the hierarchical construction is thus that of a set. On this basis, the conceptual equivalent of the Merge construction $\gamma = \{\alpha, \beta\}$ is

$$\gamma = [\textbf{set}\ \alpha\ \beta]$$

HCC can replicate Merge in this way. But the possibility of utilizing any concept as the encompassing element means HCC satisfies the requirement for unbounded generativity in more than one sense. HCC brings two or more constituents together to produce a new object. Innovation of this completely new concept then paves the way for further novel constructions in a potentially ongoing way. The process is inherently unbounded. With Merge the situation is different. Here the result of a construction is a set of existing objects, rather than an inherently new object. This is a point Boeckx (2009) particularly emphasizes. As he says, 'once you combine two units, X and Y, the output is not some new element Z, but either X or Y.'[6] On the face of it, the effect is to limit

[6]This also recalls Hinzen's (2009) point, that Merge is unable to bring about 'categorial

the generative range of Merge to the powerset of the original elements.

This is where the distinction between internal and external Merge becomes important. Internal Merge is the special case of the operator in which a merged entity is *re-merged* with one of its constituents. By virtue of the fact that this action can be repeated indefinitely, the requirement for unbounded generativity is satisfied. Since HCC subsumes Merge, it also satisfies the requirement in this way. HCC thus fulfils the requirement in two ways. Another minor difference between the operators relates to the number of allowed constituents. In HCC there can be any number of encompassed concepts, whereas Merge is normally assumed to use only two. Again, the relationship is one in which HCC is seen to generalize Merge.

With these points of comparison in mind, it is possible to move on to consider how HCC and Merge may be computationally inter-related. It is important to acknowledge straight away that there might be no functional relationship at all. The language system might use Merge to build a syntactic structure expressing a particular meaning, while the conceptual system uses HCC to build a representation of meaning in a completely independent way. Complete independence of the two operators is a viable option.

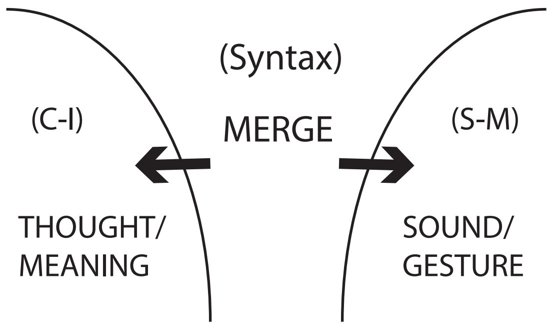

Figure 1: The standard minimalist architecture.

One attraction of the 'complete independence' interpretation is that it readily conforms to the minimalist architecture, as schematized in Figure 1 (after Hinzen, 2009, p. 126). The language system is here seen to have Syntax (mediated by Merge) at its core, with two outward-facing interfaces attached. These are represented by the two curved lines. One interface connects Syntax to C-I, the conceptual-intentional system dealing with thought and meaning. The other connects Syntax to S-M, the sensory-motor system dealing with articulation and externalization. Under this configuration, representations of meaning constructed in C-I might not use Merge in any way. It is consistent with HCC and Merge being entirely independent.

The assumption of independence has problematic implications for cognition

change' (p. 137).

more generally, however. A hierarchical syntactic structure expressing a certain meaning is essentially the same object as a hierarchical *conceptual* structure which expresses it. In both cases, leaf nodes reference primitive meanings, while internal nodes reference imposed meanings. In the syntactic structure, the nodes are content words and grammatical categories respectively, whereas in the conceptual structure, they are just concept names. Setting that difference aside, the two structures are essentially the same.

Mapping one to the other is straightforward in result. Say we have a noun phrase in the syntactic structure, comprised of a noun and definite article (e.g., *the book*). The root node of the hierarchical unit will then be labeled ART.DEF or something of the kind, while the constituent is specified as the content word *book*. The definite article is understood to impose a grammatical meaning on the referenced entity—the meaning of being a definite thing. The equivalent conceptual structure then has the concept of a definite thing as its encompassing element, and the concept of a book as the encompassed element. The notation ($_{\text{ART.DEF}}$ *book*) thus translates into [**definite.thing** book].

The complication on this relates to the ordering of branches. In syntactic structure, branches are ordered according to the grammar of the language, and different orderings may be used to convey different meanings. For example, in *John sees the book*, *John* is classified as subject of the action solely by its placement. Hierarchical conceptual structures, on the other hand, have unordered branches.[7] When translating syntactic to conceptual structure, we have to make sure any semantic information established by ordering is made explicit. Going the other way, we have to make sure any semantic information that can be dealt with by ordering is processed in this way, while the grammar of the language is also upheld. Going from syntactic to conceptual structure, we have to ensure orderings are put in correctly. Going the other way, we have to ensure they are extracted correctly.

Consider, for example, the syntactic structure of *John sees the book*:

($_{\text{VP}}$ *sees* ($_{\text{N}}$ *John*) ($_{\text{NP}}$ ($_{\text{DET}}$ *the*) ($_{\text{N}}$ *book*)))

Say we would like to convert this to the equivalent conceptual structure. Since subject and object are both established by use of SVO ordering, the conceptual structure must classify subject and object explicitly. It must include at least the classification

[**subject** John].

Since the referenced book is classified as a definite thing (by use of the definite article *the*), we also need

[**definite.thing** book].

This itself must be subject to classification as object of the phrase:

[7] Ordering of constituents is generally assumed not to play a role in conceptual structure (cf. Reinhart, 1976).

[**object** [**definite.thing** book]]

Given the seeing action referenced by *sees* encompasses both subject and object, the corresponding conceptual structure is then found to be

[**seeing.action** [**subject** John] [**object** [**definite.thing** book]]]

This builds the idea of John seeing a particular book, which is also the meaning of *John sees the book*.

For purposes of putting the conversion on an algorithmic footing, it is convenient to focus on the inverse translation. The mapping from conceptual to syntactic structure is more easily formalized. In this case, the orderings that need to be introduced are defined by the head-placement preferences of the language (e.g., preference for SVO ordering), as they apply to the utterance in question. Given a suitable way of expressing these as rules, the conversion can then be fully automated. Indeed, we can directly convert a conceptual structure into a syntactically well-formed utterance expressing the same meaning.

How does this work in practice? Consider again the conceptual structure above. As noted, this builds the idea of John seeing a particular book. For purposes of converting this into *John sees the book*, two things are required. First, we need a lexicon mapping concept names to words of corresponding meaning. Assume the following is given:

book ← book
the ← definite.thing
John ← John
sees ← seeing.action

Each rule here has a concept specification on the right, and an English symbol (i.e., word) with the corresponding meaning on the left. The symbol for the concept of a book is defined to be *book*, for example.

Also required are rules to specify correct orderings. As noted, these must deal with implicit classifications, while also enforcing relevant grammatical preferences. To correctly derive this utterance, we need to enforce (1) the preference for SVO ordering, (2) implicit classification of subject and object, and (3) the preference for head-initial organization in simple phrases. The following three rules have the required effect.

2 1 3 $\xleftarrow{\text{a}}$ [= subject object]
2 $\xleftarrow{\text{b}}$ [subject/object =]
1 2 $\xleftarrow{\text{c}}$ [= =]

Like the lexical rules, these are notated on a right-to-left basis but with labels placed over the arrows for easy reference. The first rule (labeled $\xleftarrow{\text{a}}$) enforces the preference for SVO organization; the second (labeled $\xleftarrow{\text{b}}$) enforces the preference to express subject and object classifications by ordering, while the third (labeled $\xleftarrow{\text{c}}$) enforces the preference for head-initial organization in simple phrases.

The notation works in the following way. Rule $\xleftarrow{a}$ applies to any conceptual structure of the form

[= subject object]

The structure can have anything as its encompassing element (this is what the '=' means) but the encompassed elements must *comprise* subject and object concepts.[8] A subject concept is specified either by name (i.e., as 'subject') or as a construct for which 'subject' is encompassing either explicitly or implicitly. This means [**subject** John] is a subject concept, as is [**definite.thing** [**subject** John]].

The numbers on the left of a rule specify the way symbols should be ordered. Each number indexes an element of the specification on the right, while its *position* says where symbols arising for that element should be placed. Rule $\xleftarrow{a}$ has '2 1 3' on the left. This means symbols arising for whatever matches the 2nd element should be placed first, followed by symbols for whatever matches the 1st element, followed by symbols for whatever matches the 3rd element. Given the structures it can match to, this rule has the effect of enforcing SVO ordering.

The second rule uses '/' to express alternatives. The specification in this case is considered to match any structure in which the first element is *either* a subject or object concept. Given the structures it can match to, this rule enforces the preference for expressing subject and object classifications implicitly. The designation on the left in this case is just '2', meaning only the encompassed element is symbolized. The final rule deals with any single (i.e., any concept with a single encompassed element). It specifies that symbol(s) for the encompassing element should be placed before symbol(s) for the encompassed element. Given its coverage, this rule enforces head-initial organization in simple phrases. Earlier rules also take precedence, meaning $\xleftarrow{a}$ has priority over $\xleftarrow{b}$, which has priority over $\xleftarrow{c}$.

Applying these rules to the conceptual structure, we then obtain the following processing:

→ [**seeing.action** [**subject** John] [**object** [**definite.thing** book]]]
  → [**subject** John]
    ← *John* (John)
  $\xleftarrow{b}$ *John*
  ← *sees* (seeing.action)
  → [**object** [**definite.thing** book]]
    → [**definite.thing** book]
      ← *the* (definite.thing)
      ← *book* (book)
    $\xleftarrow{c}$ *the book*
  $\xleftarrow{b}$ *the book*

[8] As this is a concept specification, encompassed elements are unordered.

$\xleftarrow{a}$ *John sees the book*

This listing shows the recursive computation that is obtained. The general effect is to break the conceptual structure down into its terminal elements. (Indentation is used to represent embedding.) At the limit of each recursive decomposition, a concept name is translated to its corresponding symbol. Then, as the recursion unwinds, the relevant ordering rule is applied at each stage. The effect off this is to incrementally assemble a sequence of symbols whose grammatical structure encodes the original conceptual structure, subject to the specified grammatical preferences. Hence, *John sees the book* is obtained as the final output.

Notice that for each application of a lexical rule, there is a line which terminates with the relevant concept name. For example, mapping of the concept name 'book' to the symbol *book* is denoted by the line

$\leftarrow$ *book* (book)

For each application of an ordering rule, there is a line showing the concept that is processed and, at the same level of indentation lower down, a second line showing the symbol sequence assembled. Use of rule $\xleftarrow{b}$ to turn [**definite.thing** book] into *the book* thus has an upper line of the form

$\rightarrow$ [**definite.thing** book]

and a lower line of the form

$\xleftarrow{b}$ *the book*

The example illustrates conversion of conceptual to syntactic structure in a simple case. More complex examples are easily generated, however. In fact, all the later examples of the previous section can be put to use in this way. We can also vary the language in which the output comes to be expressed. Consider again the utterance *John read the letter.* Translated into Japanese, this becomes

Johnga tegamio yonda.

The utterance is then a verb phrase incorporating two role-marked nouns: *John* marked as subject (SUBJ), and *tegami* marked as object (OBJ). In more detail, the grammatical structure is shown by Kuno (1973, p. 10; WALS, Ch. 82, Ex. 2) to be as follows:

John-ga	*tegami-o*	*yon-da*
John-SUBJ	letter-OBJ	read-PST

'John read the letter'

Derivation of this Japanese sentence by conversion of conceptual structure can then be demonstrated. As previously noted, a conceptual construction with the meaning of 'John read the book' is

[[**past.behavior** reading.action] [**subject** John] [**definite.thing** letter]]

To perform the conversion, we need the following lexicon mapping utilized concepts to suitable Japanese symbols:

John $\leftarrow$ John
tegami $\leftarrow$ letter
yon $\leftarrow$ reading.action
da $\leftarrow$ past.behavior
ga $\leftarrow$ subject
o $\leftarrow$ definite.thing

Each concept is mapped to the word or morpheme which has the concept as its meaning. (Strictly speaking, the rules form a morpho-lexicon.) We also need the following two ordering rules to capture the grammatical preferences of Japanese which influence this utterance.

2 3 1 $\xleftarrow{a}$ [= subject definite.thing]
2 1 $\xleftarrow{b}$ [= =]

Given the concepts it can match to, use of '2 3 1' in rule $\xleftarrow{a}$ enforces SOV organization, while rule $\xleftarrow{b}$ enforces head-final organization in simple phrases.

Conversion of the conceptual structure by application of the rules then proceeds as follows:

$\rightarrow$ [[**past.behavior** reading.action] [**subject** John] [**definite.thing** letter]]
  $\rightarrow$ [**subject** John]
    $\leftarrow$ *John* (John)
    $\leftarrow$ *ga* (subject)
  $\xleftarrow{b}$ *John ga*
  $\rightarrow$ [**definite.thing** letter]
    $\leftarrow$ *tegami* (letter)
    $\leftarrow$ *o* (definite.thing)
  $\xleftarrow{b}$ *tegami o*
  $\rightarrow$ [**past.behavior** reading.action]
    $\leftarrow$ *yon* (reading.action)
    $\leftarrow$ *da* (past.behavior)
  $\xleftarrow{b}$ *yon da*
$\xleftarrow{a}$ *John ga tegami o yon da*

With conventional word-breaks imposed, the output is then *Johnga tegamio yonda.* This is the desired Japanese sentence—it is a Japanese expression of the original meaning.

Another example taken from the previous section is

[**yesterday.event**
  [**giving.action**
    [**subject** [**indefinite.thing** man]]

[**object** bread]
[**indirect.object** John]]]

This builds the idea of an indefinite man giving bread to an individual, John, at a particular point in time, namely yesterday. The meaning is what we would express in English by saying something like *Yesterday, a man gave bread to John.* A sentence from the Suriname language of Arawak with a not dissimilar meaning is

Miaka aba wadili sika khali damyn.

This translates as ‘Yesterday a man gave cassava bread to me.’ The grammatical structure is analyzed by Pet (1987; see also WALS, Ch. 84, Ex. 4) as follows.

Miaka	*aba*	*wadili*	*sika*	*khali*	*da-myn*
yesterday	INDEF	man	give	cassava.bread	1SG-to

‘Yesterday a man gave cassava.bread to me’

To capture the meaning of this Arawak sentence, the conceptual structure above needs to be modified in two ways. The recipient of the action needs to be specified as ‘me’ rather than ‘John’, and the object should be ‘cassava.bread’ rather than ‘bread’. This yields

[**yesterday.event**
  [**giving.action**
    [**subject** [**indefinite.thing** man]]
    [**object** cassava.bread]
    [**indirect.object** me]]]

Generation of the Arawak sentence by conversion of conceptual structure can then be demonstrated. A morpho-lexicon mapping the utilized concepts to suitable Arawak symbols is as follows:

khali ← cassava.bread
sika ← giving.action
aba ← indefinite.thing
myn ← indirect.object
wadili ← man
da ← me
miaka ← yesterday.event

We also need the following four ordering rules to capture the grammatical preferences which influence this utterance. The first captures the preference for SVO organization; the second, the preference for dealing with subject and object classifications by ordering; the third, the preference for head-final organization in simple phrases denoting an indirect object; and the last, the preference for head-final organization otherwise.

2 1 3 4 $\xleftarrow{a}$ [= subject object indirect.object]
2 $\xleftarrow{b}$ [subject/object =]
2 1 $\xleftarrow{c}$ [indirect.object =]
1 2 $\xleftarrow{d}$ [= =]

Applying these rules to the conceptual structure then produces the following processing. (Some lines are truncated.)

$\rightarrow$ [**yesterday.event** [**giving.action** [**subject** [**indefinite.thing** man]] ...
$\leftarrow$ *miaka* (yesterday.event)
$\rightarrow$ [**giving.action** [**subject** [**indefinite.thing** man]] [**object** ...
$\rightarrow$ [**subject** [**indefinite.thing** man]]
$\rightarrow$ [**indefinite.thing** man]
$\leftarrow$ *aba* (indefinite.thing)
$\leftarrow$ *wadili* (man)
$\xleftarrow{d}$ *aba wadili*
$\xleftarrow{b}$ *aba wadili*
$\leftarrow$ *sika* (giving.action)
$\rightarrow$ [**object** cassava.bread]
$\leftarrow$ *khali* (cassava.bread)
$\xleftarrow{b}$ *khali*
$\rightarrow$ [**indirect.object** me]
$\leftarrow$ *da* (me)
$\leftarrow$ *myn* (indirect.object)
$\xleftarrow{c}$ *da myn*
$\xleftarrow{a}$ *aba wadili sika khali da myn*
$\xleftarrow{d}$ *miaka aba wadili sika khali da myn*

With standard word-breaks imposed, the final output is *Miaka aba wadili sika khali damyn.* This is the Arawak sentence expressing the original meaning.

To complete this series of examples, it is also useful to look at

[**question**
[[**drinking.action** focal.thing]
[**subject** [definite.thing teacher]]
[**object** [**definite.thing** [**substance** water]]]]]

Recall that this builds an idea that would be expressed in English by asking the question 'Is the teacher drinking the water?' Imagine we would like to derive this utterance in German by conversion of conceptual structure. A suitable morpho-lexicon for the task is

lehrer $\leftarrow$ teacher
wasser $\leftarrow$ water

trink ← drinking.action
das ← [**definite.thing** substance]
der ← definite.thing
t ← focal.thing

Note the use of a structured specification in the case of *das*, to capture the restricted range of this determiner. The following four ordering rules are also needed to capture relevant grammatical preferences of German. The first captures the preference to deal with subject and object classifications by ordering alone; the second, the preference for VSO structure given a meaning classified as a question; the third, the preference for SVO organization otherwise, and the last, the preference for head-initial organization in simple phrases.

2 $\xleftarrow{a}$ [subject/object =]
2 3 4 $\xleftarrow{b}$ [question [= subject object]]
2 1 3 $\xleftarrow{c}$ [= subject object]
1 2 $\xleftarrow{e}$ [= =]

The processing obtained is then as follows:

→ [**question** [[**drinking.action** focal.thing] [**subject** ...
  → [**drinking.action** focal.thing]
    ← *trink* (drinking.action)
    ← *t* (focal.thing)
  $\xleftarrow{e}$ *trink t*
  → [**subject** [**definite.thing** teacher]]
    → [**definite.thing** teacher]
      ← *der* (definite.thing)
      ← *lehrer* (teacher)
    $\xleftarrow{e}$ *der lehrer*
  $\xleftarrow{a}$ *der lehrer*
  → [**object** [**definite.thing** [**substance** water]]]
    ← *das* ([definite.thing substance])
    ← *wasser* (water)
  $\xleftarrow{a}$ *das wasser*
$\xleftarrow{b}$ *trink t der lehrer das wasser*

With standard word-breaks imposed, the output is *Trinkt der lehrer das wasser*. This is the desired German utterance. It is the question 'Is the teacher drinking the water?' expressed in German.

If the intended meaning is *not* classified as a question, only the inner conceptual structure remains:

[[**drinking.action** focal.thing]
  [**subject** [definite.thing teacher]]
  [**object** [**definite.thing** [**substance** water]]]]

This builds a meaning we would express in English by saying 'the teacher is drinking the water.' Applying the rules to this reduced structure then invokes the default SVO ordering, as follows:

$\rightarrow$ [[**drinking.action** focal.thing] [**subject** ...
$\rightarrow$ [**subject** [**definite.thing** teacher]]
$\rightarrow$ [**definite.thing** teacher]
$\leftarrow$ *der* (definite.thing)
$\leftarrow$ *lehrer* (teacher)
$\xleftarrow{e}$ *der lehrer*
$\xleftarrow{a}$ *der lehrer*
$\rightarrow$ [**drinking.action** focal.thing]
$\leftarrow$ *trink* (drinking.action)
$\leftarrow$ *t* (focal.thing)
$\xleftarrow{e}$ *trink t*
$\rightarrow$ [**object** [**definite.thing** [**substance** water]]]
$\leftarrow$ *das* ([definite.thing substance])
$\leftarrow$ *wasser* (water)
$\xleftarrow{a}$ *das wasser*
$\xleftarrow{c}$ *der lehrer trink t das wasser*

With standard word-breaks imposed, the output is then found to be *der lehrer trinkt das wasser*. This is 'the teacher is drinking the water' in German.

The outlook for conversion of conceptual structure is not unpromising then. If these cases are any guide, syntactically well-formed utterances can be derived directly from conceptual structure, given suitable lexical and ordering rules. Since the conversion can be made in either direction, it follows that conceptual and syntactic structure are computationally interchangeable. Syntactic structure can be used as a substitute for conceptual structure, and vice versa.

Is it credible the mind would fail to take advantage of this? If the products of HCC can be derived from the products of Merge (and vice versa), what would be achieved by having independent operators? If the language system provides Merge, it seems likely the conceptual system would use this for purposes of HCC, thus avoiding duplication of effort. Likewise, if the conceptual system provides HCC, it seems likely the language system would use this for purposes of Merge. Two general arrangements then suggest themselves: a 'syntax-first' arrangement based on Merge, and a 'semantics-first' arrangement based on HCC. Up to a point, each is just an inversion of the other; but the architectural implications differ significantly. Under the syntax-first arrangement, there is no requirement for an implementation of HCC. Under the semantics-first arrangement, there is no requirement for syntactic rules. Rules covering grammatical preferences suffice.

With regard to the minimalist architecture, the two variants of Figure 2 then suggest themselves. The interface between Syntax and C-I might run either left-to-right or right-to-left, with the latter applying in the case of syntax-first

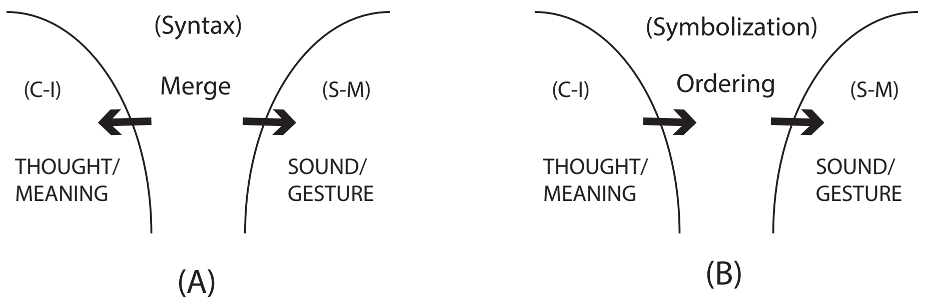

Figure 2: (A) Syntax-first processing; (B) Semantics-first processing.

processing, and the former in the case of semantics-first processing. Under the syntax-first configuration, the conceptual system gets hierarchical construction done 'for free.' HCC is resourced by means of linguistic Merge, and the conceptual system is simplified accordingly. Under semantics-first, it is the other way around: Syntax gets constructive services for free. Syntax is seen to fulfil the purposes of Merge by means of HCC. The two arrangements also give rise to different forms of internal communication. In version (A), we have the standard setup, in which Syntax sends structures to both C-I and S-M. In version (B) we have the semantics-first alternative, in which C-I sends structures to an intervening symbolization module, which then uses ordering rules to produce syntactically well-formed outputs. Examples such as the ones set out above suggest both arrangements are viable.

4 Discussion

Concepts can be assembled hierarchically by means of language, and for this purpose, language appears perfectly designed. This much is generally agreed (Pinker, 1994; Jackendoff, 2002; Chomsky, 2007a). Does it make any difference if concepts are *inherently* hierarchically combinable? The possibility then arises that the hierarchical structures of language might be expressing the inherently hierarchical structures of conceptualization. But is it credible the language system is organized in this way? To some degree, this is just a new way of raising the issue of how thought and language are related. It brings us back to the fundamental question about which process is in charge (cf. Chater and Christiansen, 2010). Is language the shaping influence over thought, or does thought shape language? Is language a means of accomplishing conceptualization, or merely a way of accommodating it?

Hinzen puts the question like this:

> is syntax the dress or the skeleton of thought? Is syntactic complexity a contingent way of dressing up human thought, viewed as

> something independent from language, in a linguistic guise? Or is syntax what literally *constructs* a thought and gives it its essential shape? (Hinzen, 2009, p. 128)

The observation that concepts are hierarchically combinable falls short of providing an answer. It does, however, provide a novel way of addressing the question. Given a suitable notation, it is possible to examine hierarchical conceptualization in a formal way, abstracting away the role of language altogether. What becomes apparent is that this is an inherently recursive, unboundedly generative operation in its own right. Its products remain semantically precise regardless of their structural complexity. They also exhibit a semantic variety that seems no less broad than what is observed in language. A mechanistic relationship between hierarchical construction in the conceptual system and hierarchical construction in the language system is then on the cards. HCC might be resourced by Merge, or it could be the other way around.

Unfortunately, even with the computational aspects of this alternation taken into account, the question about whether syntax is the dress or skeleton of thought remains hard to answer. Both the syntax-first arrangement (syntax as 'skeleton') and the semantics-first arrangement (syntax as 'dress') seem to be computationally viable. We cannot choose between them purely on the basis of tractability. Is there some other way of doing so? Can the literature be brought to bear in any way? Theorists working in the minimalist tradition often assume an intimate relationship between Merge and thought, and this may be described in terms that cast thought as the prior process. Hauser, for example, argues that we should see language as a 'mind-internal computational system designed for thought and often externalized in communication' (Hauser, 2009, p 74). This seems consistent with a semantics-first arrangement, and might be viewed as a vote in favour. But that would be a misreading. The semantics-first arrangement eliminates syntactic rules altogether, which is more than Hauser intends. What Hauser is emphasizing is not the degree to which conceptualization can be a hierarchically constructive process in its own right, but the degree to which syntax can be its medium.

Another theorist who emphasizes the priority of thought, while stopping short of the semantics-first interpretation, is Hinzen (2009). Hinzen goes so far as to argue that thought must have a generative system that powers it, and that this must obey the same functional principles as Merge. But, again, this is not taken to imply that conceptual construction might take the place of syntax. As Hinzen says,

> thought is as generative and discretely infinite as language is ... such productivity is only possible if there is a generative system behind thought that powers it. Could that system really employ radically different generative principles than the ones we now think the computational system of language (syntax) exhibits? ... If Merge, which is thought to be the basic computational operation of human syntax, is what is minimally needed to get a system with the basic properties of language, could it *fail* to exist in another system, the system

> of "thought," that exhibits these very properties as well?' (Hinzen, 2009, p. 128)

Again, this might be read as support for the semantics-first architecture. But, as with Hauser, what Hinzen is suggesting is that the generative system of thought and the generative system of language are essentially one and the same. In Hinzen's view, 'the generative system of language underlies and is actually indistinguishable from the generative system that powers abstract thought' (Hinzen, 2009, p. 125).

Like Hauser and Hinzen, Chomsky also argues that 'language evolved, and is designed primarily as an instrument of thought' (Chomsky, 2009, p. 29), Again, this is not taken as tending towards a semantics-first account. On the other hand, Chomsky does come closer to this in the position he takes on the evolution of Merge.[9] In Chomsky's view, the first human to acquire this capability was

> instantly endowed with intellectual capacities far superior to those of others, transmitted to offspring and coming to predominate, perhaps linked as a secondary process to the [sensory-motor] system for externalization and interaction, including communication as a special case. (Chomsky, 2005, p. 12)

This at least admits the possibility of a purely conceptual version of Merge, driving thought alone. The existence of such a mechanism, in the form of HCC, is key to the semantics-first account. But there is no suggestion from Chomsky that structured conceptualization might stand in for Merge. Elimination of syntax is not part of the proposal. Like Hauser and Hinzen, Chomsky sees syntax and the generative process of thought as essentially the same thing, while acknowledging the latter might have existed independently at some point in human evolution (cf. Chomsky, 2009, 2007b, 2012). Chomsky generally rejects the possibility of there being any hierarchically generative system other that Merge, commenting in one case that 'To date, I am not aware of any real examples of unbounded Merge apart from language' (Chomsky, 2007a, p. 20).

The view that Chomsky, Hauser and Hinzen take on this issue is not unrepresentative of the minimalist literature more generally. While thought is taken as prior in some sense, it is also seen to be satisfactorily mediated by syntax, and thus by Merge. No theorist advocates wholesale elimination of syntax within the explanation of language. The weight of evidence from the literature strongly favours the syntax-first account, then.

Nevertheless, this account is not without problems. One issue is the assumption of there being no examples of unbounded Merge other than language. In light of the hierarchical combinability of concepts, this is incorrect: HCC is just such an example. Another problem is the assumption of syntax being a satisfactory medium of thought. This implies that different linguistic constructions should realize different thoughts, which seems open to question. The

[9]The assumption that 'Merge or some equivalent is a minimal requirement [for language]' (Chomsky, 2005, p. 11-12) is also relevant here, HCC being just such an equivalent.

same thought can often be expressed in different ways. A third issue relates to what Chomsky calls ‘the curious but ubiquitous phenomenon of displacement in natural language’ (Chomsky, 2009, p. 31). If syntax is a medium of thought, there seems no reason why purely linguistic phenomena like this should exist.

The semantics-first account is at an advantage in such cases. It acknowledges the existence of HCC as an equivalent of Merge, and is consistent with conceptual constructions having multiple linguistic expressions. It also naturally accommodates phenomena of displacement and agreement. In the latter, we have multiple symbols encoding a single meaning. This can be explained as exploitation of redundancy, a strategy with significant benefits in any encoding task. The former involves syntactic entities being displaced away from the position they seem to require on grounds of grammatical structure. In the semantics-first argument, the assumption that symbols take their place according to grammatical structure alone is eliminated.

There are explanatory pros and cons on both sides, then. It has to be acknowledged that there is insufficient evidence to justify any final conclusion about the relationship between Merge and HCC. The possibility of there being no functional connection at all cannot be ruled out. It might be that the conceptual system pursues hierarchical construction using more or less the same operator as the language system, but in a completely independent way. This presupposes significant duplication of effort in the mind, however.

If complete independence is ruled out, what remains is either the syntax-first or semantics-first forms of organization. In favour of the former is the fact that it is more consistent with prevailing views on how thought and language are woven together. In favour of the latter is the fact that it resolves some problems with the syntax-first account. It is also potentially more parsimonious. Syntactic rules treat hierarchical construction and symbol ordering as a single process. A syntactic account thus has to treat every way of conjoining hierarchical construction and ordering as a special case. The semantics-first alternative separates the two factors, and makes one the responsibility of the conceptual system. Within this divide-and-conquer approach, the explanation of language is potentially simplified. Clearly, more research is required to bring more clarity to the situation. It is hoped that future work will make further progress towards that goal.

References

Boeckx, C. (2009). The Nature of Merge: Consequences for Language, Mind, and Biology. In Piattelli-Palmarini, Uriagereka and Salaburu (Eds.), *Of Minds and Language: A dialogue with Noam Chomsky in the Basque Country* (pp. 44-57), Oxford University Press.

Chater, N. and Christiansen, M. H. (2010). Language Acquisition Meets Language Evolution. *Cognitive Science*, *34*, No. 7 (pp. 1131-1157).

Chomsky, N. (1995). *The Minimalist Program*, The MIT Press.

Chomsky, N. (2001). Derivation by Phase. *Ken Hale: A Life in Language* (pp. 1-52), The MIT Press.

Chomsky, N. (2005). Three Factors in Language Design. *Linguistic Inquiry*, *36* (pp. 1-22).

Chomsky, N. (2007a). Of Minds and Language. *Biolinguistics*, *1* (pp. 9-27).

Chomsky, N. (2007b). Biolinguistic Explorations: Design, Development, Evolution. *International Journal of Philosophical Studies*, *15* (pp. 1-16).

Chomsky, N. (2009). Opening Remarks. In Piattelli-Palmarini, Uriagereka and Salaburu (Eds.), *Of Minds and Language: A dialogue with Noam Chomsky in the Basque Country* (pp. 13-43), Oxford University Press.

Chomsky, N. (2012). *The Science of Language: Interviews with James McGilvray*, Cambridge: Cambridge University Press.

Costello, F. J. and Keane, M. T. (2001). Testing Two Theories of Conceptual Combination: Alignment versus Diagnosticity in the Comprehension and Production of Combined Concepts. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *27*, No. 1 (pp. 255-271).

Dryer, M. S. and Haspelmath, M. (eds.) (2011). *The World Atlas of Language Structures Online*, Munich: Max Planck Digital Library. Package online at http://wals.info/ Accessed on 2013-04-06.

Hampton, J. A. (1991). The Combination of Prototype Concepts. In Schwanenflugel (Ed.), *The Psychology of Word Meanings*, Hillsdale, N.J: Lawrence Erlbaum Associates.

Hampton, J. (1997). Conceptual Combination: Conjunction and Negation of Natural Concepts. *Memory and Cognition*, *25* (pp. 888-909).

Hampton, J. (2011). Conceptual Combination and Fuzzy Logic. In Bělohlavek and Klir (Eds.), *Concepts and Fuzzy Logic* (pp. 209-232), MIT Press.

Hauser, M. D., Chomsky, N. and Fitch, W. T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, *198* (pp. 15691579).

Hauser, M. D. (2009). Evolingo: The Nature of the Language Faculty. In Piattelli-Palmarini, Uriagereka and Salaburu (Eds.), *Of Minds and Language: A dialogue with Noam Chomsky in the Basque Country* (pp. 74-84), Oxford University Press.

Hinzen, W. (2009). Hierarchy, Merge, and Truth. In Piattelli-Palmarini, Uriagereka and Salaburu (Eds.), *Of Minds and Language: A dialogue with Noam Chomsky in the Basque Country* (pp. 124-141), Oxford University Press.

Jackendoff, R. (2002). *Foundations of Language: Brain, Meaning, Grammar, Evolution*, Oxford University Press.

Kuno, S. (1973). *The Structure of the Japanese Language*, Cambridge, Mass: MIT Press.

McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P. and Levin, M. I. (1985/1962). *LISP 1.5 Programmer's Manual (2nd ed.)*, Cambridge, Mass: MIT Press.

Murphy, G. L. (2002). *The Big Book of Concepts*, London, England: The MIT Press.

Pet, W. A. P. (1987). *Lokono Dian, the Arawak Language of Surniname: A Sketch of its Grammatical Structure and Lexicon*, Cornell University Doctoral Thesis.

Pinker, S. (1994). *The Language Instinct: The new Science of Language and Mind*, The Penguin Press.

Reinhart, T. (1976). *The Syntactic Domain of Anaphora, Doctoral dissertation*, Cambridge, Massachusets: MIT.

Rips, L. J. (1995). The Current Status of Research on Concept Combination. *Mind and Language*, *10* (pp. 72-104).

Sowa, J. F. (1984). *Conceptual Structures: Information in Mind and Machine*, Reading, Mass: Addison-Wesley.

Thagard, P. (1997). Coherent and Creative Conceptual Combination. In Ward, Smith and Viad (Eds.), *Creative Thought: an Investigation of Conceptual Structures and Processes*, Washington, DC: American Psychological Association.

Wisniewski, E. J. (1997). When Concepts Combine. *Psychonomic Bulletin and Review*, *4* (pp. 167-183).